Detección de Manipulación en Ficheros Multimedia Basado en Algoritmos de Compresión

TRABAJO FIN DE GRADO
GRADO EN INGENIERÍA INFORMÁTICA
CURSO 2017 – 2018

Javier Martín-Pozuelo Salvador

Directores
Luis Javier García Villalba
Ana Lucila Sandoval Orozco

Departamento de Ingeniería del Software e Inteligencia Artificial
Facultad de Informática
Universidad Complutense de Madrid

Madrid, Septiembre de 2018

Agradecimientos

Quiero dar las gracias a los directores de este trabajo Luis Javier García Villalba y Ana Lucila Sandoval Orozco por haberme dado la oportunidad de participar en este proyecto y haberme guiado durante la realización del mismo.
También quiero agradecer a Esteban Armas Vega por ayudarme en todo momento respondiéndome a todas las dudas que me han surgido.
Por último, dar las gracias a mi familia y amigos que me han dado ánimos durante todos estos meses.

Índice General

Índice de Figuras

Índice de Tablas

Índice de Algoritmos

Abstract

Digital images and videos play an important role in everyday life. Today, the majority of the population owns state-of-the-art cameras integrated into their mobile device. Technological development not only facilitates the generation of multimedia content, but also the intentional manipulation of it, and this is where forensic techniques of detecting manipulation on images and videos take on great importance. In this work two forensic methodologies based on compression algorithms are proposed: The first one tries to detect the presence of recompression in a digital video by means of the analysis of its macroblocks, characteristic of the H.264-MPEG4 standard. Subsequently, the vectorial support machine is used to create the model that allows the verification of the number of recompressions of a video. The second methodology explained in this work aims to detect splicing alterations, i.e. regions that do not belong to the original content of a digital image, a technique based on the error rate introduced by the JPEG compression algorithm each time it recompresses an image.

Keywords: Forensic Analysis, Manipulation, Classification, Compression, Digital Images, Digital Videos, Vector Support Machine, SVM, JPEG, Splicing, Error Level Analysis, ELA, H.264, MPEG4, Macroblocks, P-Frames, Motion Vectors.

Resumen

Las imágenes y vídeos digitales juegan un papel muy importante en la vida cotidiana. A día de hoy, la mayor parte de la población es poseedora de cámaras fotográficas de última generación integradas en su dispositivo móvil. El desarrollo tecnológico no sólo facilita la generación de contenido multimedia, sino también la manipulación intencionada de éste, y es aquí donde las técnicas forenses de detección de manipulación sobre imágenes y vídeos cobran gran importancia. En este trabajo se proponen dos metodologías forenses basadas en algoritmos de compresión: La primera de ellas trata de detectar la presencia de recompresión en un vídeo digital mediante el análisis de sus macrobloques, característica propia del estándar H.264-MPEG4. Posteriormente, se utiliza la máquina de soporte vectorial para crear el modelo que permita la verificación del número de recompresiones de un vídeo. La segunda metodología que se explica en este trabajo tiene por objetivo detectar alteraciones de tipo 'empalme', es decir, regiones que no pertenecen al contenido original de una imagen digital, técnica que está basada en la tasa de error que introduce el algoritmo de compresión JPEG cada vez que recomprime una imagen.

Palabras clave: Análisis Forense, Manipulación, Clasificación, Compresión, Imágenes Digitales, Vídeos Digitales, Máquina de Soporte Vectorial (SVM), JPEG, Empalme, ELA, H.264, Moving Picture Experts Group (MPEG), Macrobloques, Fotogramas P, Vectores de Movimiento.

Lista de Acrónimos

BMP	Mapa de Bits
CRF	Función de Respuesta de la Cámara
DCT	Transformada Discreta del Coseno
ELA	Error Level Analysis
FFT	Transformada de Fourier
GOP	Grupo de Imágenes
HSV	Hue-Saturation-Value
JPEG	Joint Photographic Experts Group
MBM	Modo de Macrobloque
MIT	Massachusetts Institute of Technology
MPEG	Moving Picture Experts Group

PCA	Análisis de Componentes Principales
PNG	Portable Network Graphics
PRNU	Patrón de Ruido de Respuesta no Uniforme
RBF	Función de Base Radial
RGB	Red-Green-Blue
SIFT	Scale-Invariant Feature Transform
SPN	Patrón de Ruido del Sensor
SURF	Speeded Up Robust Features
SVD	Descomposición en Valores Singulares
SVM	Máquina de Soporte Vectorial
VM	Vector de Movimiento
WT	Transformada Wavelet

Capítulo 1

Introducción

1.1. Motivación

Desde siglos atrás, el ser humano siempre ha utilizado la imagen para plasmar la realidad que le rodeaba, o modificarla, en función del mensaje que se quisiera transmitir. Aunque esta evolución, sin duda, tiene un antes y un después con la creación de la fotografía en el siglo XIX.

> "La excitación que acompañó a la invención de la fotografía fue la sensación de que el hombre por primera vez podía ver el mundo como realmente era"(Collier 1986: 3) [Nie].

Esta afirmación que hace Collier acerca de la fotografía podría no ajustarse al pie de la letra en la actual era digital. Actualmente existe un significante número de delitos informáticos relacionados con la posesión ilicita, distribución o modificación de contenido multimedia. El presunto uso de dispositvos móviles para este propósito hace que estos dispositivos sean una importante fuente de evidencia, hecho por el cual los análisis forenses deben ser capaces de autenticar el contenido y examinar si es original o fue manipulado.

No obstante, la manipulación de contenido visual no ha sido algo exclusivo de la era digital actual. A lo largo del tiempo la manipulación siempre ha estado presente:

- En pintura, se ha retocado la imagen que se quería transmitir al público, por ejemplo, En 'El Juicio Final', el pintor Miguel Ángel cubrió la desnudez de algunas figuras a posteriori por orden del Papa.
- En fotografía convencional, era posible la manipulación mediante empalmes de los negativos de las fotografías, por ejemplo en la Figura 1.1 la famosa foto del dictador soviético Iósif Stalin junto con su comisario para Asuntos Internos, el cual desparece de la foto por orden de Stalin tras ser ejecutado en 1940.
- En vídeo, la manipulación se basaba en unir secuencias de imágenes, también llamados fotogramas, que pertenecían a intervalos temporales distintos. Ejemplo de esto fue el Cine Fantástico de George Méliès [Nor02].

Figura 1.1: Ejemplo de Manipulación en Fotografía

Si bien antes manipular contenido visual tenía principalmente una motivación política, religiosa o cultural, a día de hoy, a parte de utilizar la manipulación con fines maliciosos, la motivación más generalizada es la publicitaria o estética, por ejemplo la Figura 1.2.

Figura 1.2: Ejemplo de Manipulación en Imagen Digital

La facilidad para manipular imágenes y vídeos digitales se ha incrementado y va en aumento en los últimos tiempos y está al alcance del usuario convencional mediante programas como Adobe Photoshop, GIMP, Adobe Premiere, etc. Incluso estas manipulaciones ya las hace de manera automática nuestro dispositvo móvil mediante nuevas herramientas que hacen uso de la inteligencia artificial como pueden ser los embellecedores de rostros, cambios de la expresión facial, mejora de la iluminación de la escena, etc.

En Julio del año 2017 los investigadores de la revista Cognitive-Research [NWW17] utilizaron un dataset de 40 escenas, 30 de las cuales fueron sometidas a cinco tipos diferentes de manipulación, incluyendo manipulaciones físicamente plausibles y no plausibles. Se mostraron a 707 participantes con el fin de evaluar la capacidad de las personas para detectar escenas manipuladas del mundo real. El estudio encontró que sólo el 60 % de las personas fue capaz de detectar las escenas falsas, e incluso entonces, sólo un 45 % de ellos fueron capaces de decir dónde exáctamente se encontraba la alteración del contenido (ver Figura 1.3). En palabras del Dr. Watson, coautor del estudio, explicó:

> "Las imágenes tienen una poderosa influencia en nuestros recuerdos, así que, si las personas no pueden diferenciar entre detalles reales y falsos en escenas, las manipulaciones podrían alterar con frecuencia lo que creemos y recordamos."

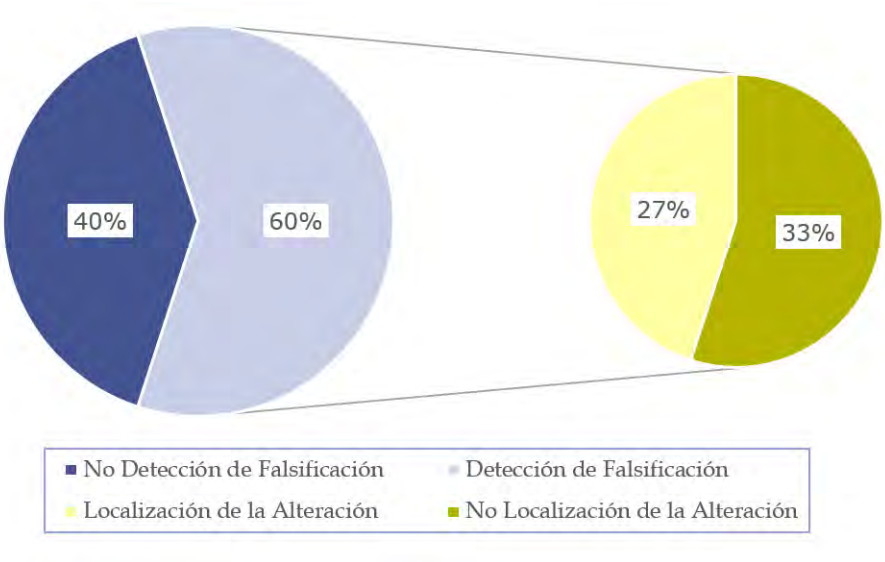


Figura 1.3: Resultados del estudio de la revista Cognitive-Research.

Por todo ello, es aquí donde entran las técnicas de detección de manipulación en imágenes y vídeos, imprescindibles para dicho fin. Se hace necesaria la revisión de los métodos de verificación de la autenticidad e integridad del contenido de una imagen o vídeo, así como la búsqueda de unos nuevos que se amolden a una realidad en constante cambio y a los problemas que en el futuro puedan plantearse.

1.2. Contexto

El presente Trabajo Fin de Grado se enmarca dentro de un proyecto de investigación titulado RAMSES, aprobado por la Comisión Europea dentro del Programa Marco de Investigación e Innovación Horizonte 2020 (Convocatoria H2020-FCT-2015, Acción de Innovación, Número de Propuesta: 700326) y en el que participa el Grupo GASS del Departamento de Ingeniería del Software e Inteligencia Artificial, integrado en la Facultad de Informática de la Universidad Complutense de Madrid (Grupo de Análisis, Seguridad y Sistemas, http://gass.ucm.es, grupo 910623 del catálogo de grupos de investigación reconocidos por la UCM).

Además de la Universidad Complutense de Madrid participan las siguientes entidades:

- Treelogic Telemática y Lógica Racional para la Empresa Europea SL (España)
- Ministério da Justiça (Portugal)
- University of Kent (Reino Unido)
- Centro Ricerche e Studi su Sicurezza e Criminalità (Italia)
- Fachhochschule fur Offentliche Verwaltung und Rechtspflege in Bayern (Alemania)
- Trilateral Research & Consulting LLP (Reino Unido)
- Politecnico di Milano (Italia)
- Service Public Federal Interieur (Bélgica)
- Universitaet des Saarlandes (Alemania)
- Dirección General de Policía - Ministerio del Interior (España)

1.3. Objetivos

Los objetivos que se han marcado en el presente trabajo son los siguientes:

- Revisar la literatura actual respecto a los usos de imágenes y vídeos falsificados para obtener un conocimiento detallado de las técnicas de manipulación usadas hoy en día.
- Analizar el estado del arte sobre las técnicas de detección de manipulaciones de imágenes y vídeos digitales.
- Diseñar e implementar un algoritmo robustos y eficiente que detecte alteriaciones de tipo empalme en imágenes digitales.

- Diseñar e implementar un algoritmo robusto y eficiente que detecte doble compresión en vídeos digitales.
- Diseñar e implementar un algoritmo robusto y eficiente que detecte el número recompresiones de un vídeo digital.
- Evaluar los algoritmos propuestos con diversos datasets.

1.4. Plan de Trabajo

Este trabajo está dividido en 6 fases detalladas a continuación:

- Búsqueda de información sobre las técnicas de manipulación existentes tanto para imágenes digitales como para vídeos así como las técnicas de detección de esas manipulaciones, en base a trabajos acádemicos anteriores que se encuentran enmarcados en este mismo área de investigación.
- Realización de la memoria paralelamente y a lo largo de todas las fases del proyecto para documentar todos los avances que se han ido realizando.
- Revisiones semanales para revisar el avance del proyecto y concretar las siguientes tareas o modificaciones a realizar.
- Preparación del entorno de trabajo mediante la instalación del Software necesario en el equipo donde se va a llevar a cabo la investigación, adaptando siempre el entorno a las exigencias naturales que van surgiendo a medida que éste avanza.
- Diseño y desarrollo de los algoritmos que se presentan en este trabajo con los cuales se pretenden alcanzar los objetivos que motivan este trabajo.
- Y por último, realización de experimentos seleccionando datasets para realizar un testing sobre los algoritmos propuestos, evaluar su eficacia, y en base a los resultados poder sacar conclusiones acerca de su idoneidad.

1.5. Estructura del Trabajo

El presente trabajo está compuesto por 6 capítulos:

- El capítulo 1 en el que se encuentra enmarcada la introducción a este trabajo y donde se describe la motivación, los objetivos, el plan de trabajo y su estructura.
- El capítulo 2 que detalla las técnicas de manipulación en imágenes y vídeos digitales con ejemplos para evidenciar su utilidad práctica.
- El capítulo 3 donde se describe el estado del Arte, es decir, las técnicas empleadas en la detección de las manipulaciones explicadas en el capítulo 2.

- El capítulo 4 en el que se presentan las contribuciones de este trabajo explicando los algoritmos desarrollados y los conceptos necesario para comprenderlos.
- El capítulo 5 que muestra los experimentos realizados con los algoritmos desarrollados en el capítulo 4 y los resultados obtenidos de los mismos.
- El capítulo 6 es el último capítulo del trabajo, y en él se recogen las conclusiones y se proponen trabajos futuros en este campo.

Capítulo 2

Técnicas de Manipulación en Imágenes Digitales y Vídeos

Este capítulo tiene como objetivo explicar las principales técnicas existentes que se utilizan a la hora de manipular el contenido multimedia de imágenes y vídeos. Además de cada técnica se muestra un ejemplo gráfico. Al final de este capítulo se presenta un estudio comparativo con las herramientas software de edición de imagen y video más utilizadas actualmente.

2.1. Técnicas de Manipulación en Imágenes Digitales

2.1.1. Copia-Pega

La manipulación 'Copia-Pega' típicamente se realiza con el objetivo de hacer que un objeto 'desaparezca' de la imagen original cubriéndolo con un pequeño fragmento copiado de otra parte de la misma imagen. Este método también se usa para duplicar objetos existentes en la imagen. Como estos bloques copiados provienen de la misma imagen todas sus caracteristicas serán compatibles con el resto del contenido por lo cual se hace muy dificil su detección por el ojo humano.

Cuando se pega la región copiada se suele acompañar del efecto 'blurring' o 'emborronado' usado generalmente sobre los bordes de la región modificada para así disminuir las irregularidades entre la región original y pegada.

Las técnicas de detección de este tipo de manipulación se centran en la búsqueda de áreas duplicadas aunque si se combina con otras técnicas de post-procesamiento, como la aplicación de filtros de color, puede dificultar bastante su detección mediante las técnicas existentes.

En la Figura 2.1 se muestra un ejemplo de esta técnica. En la Figura manipulada 2.1(b) se han duplicado los dos animales que aparecían en la Figura 2.1(a).

(a) *Imagen original*

(b) *Imagen manipulada con la técnica Copia-Pega*

Figura 2.1: Ejemplo de Manipulación con la Técnica Copia-Pega

2.1.2. Empalmes

Esta manipulación es similar a la técnica 'Copia-Pega', pero con la diferencia de que el fragmento que se copia no pertenece a la misma imagen, si no a otra distinta, es decir, la imagen manipulada es el resultado de la mezcla de dos o más imágenes. El objetivo de esta técnica es el de insertar elementos que no estaban en la escena que fue capturada originalmente. Por regla general, el bloque de imagen 'donante' ha podido ser adquirido por otro dispositivo móvil y por tanto sus caracterticas y rastros serán diferentes al resto de la imagen. Sus técnicas de detección se centran en encontrar estas variaciones en los rastros adicionales que tendrá la imagen resultado con respecto al resto del contenido de la imagen original. En la Figura 2.2 se muestra un ejemplo de esta técnica. La Figura 2.2(a) es la imagen donante, el faro es copiado y pegado en la imagen receptora Figura 2.2(b). Como resultado, se genera el empalme, la Figura 2.2(c).

2.1.3. Aplicación de Filtros

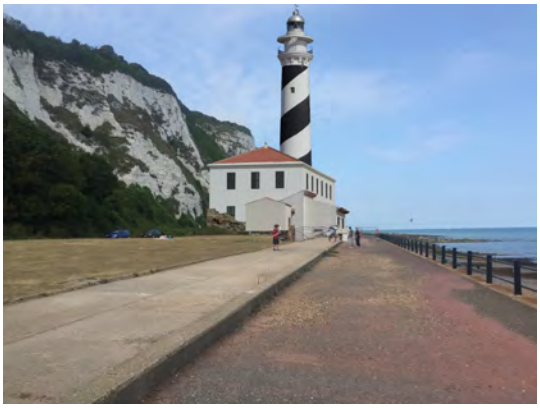
Esta manipualción es de las más utilizadas por su sencillez. Casi todos los softwares de edición de imágenes digitales incorporan una selección de filtros ya predefinidos para aplicarlos automáticamente sobre la imagen. Tiene el objetivo de mejorar el acabado final de la misma modificando aspectos como pueden ser los tonos, saturaciones, brillos, contrastes, etc.

No tienen porqué conllevar un cambio 'malicioso' en el contenido de una imagen pero se menciona en este trabajo porque es importante tenerlo en cuenta durante la detección de cualquier otra manipulación, ya que es probable que pueda afectar al funcionamiento del método empleado.

En la Figura 2.3 se muestra un ejemplo de esta técnica, donde se ha aplicado el filtro predefinido 'Sepia' del software de edición GIMP.

(a) *Imagen donante original*

(b) *Imagen receptora original*



(c) *Imagen manipulada con la técnica Empalme*

Figura 2.2: Ejemplo de Manipulación con la Técnica Empalme

2.1.4. Retoque Estético

Esta manipulación consiste en aplicar pequeñas modificaciones sobre la imagen original sin copiar ningún área del resto de la imagen o tormarla de una diferente.

Tiene como objetivo perfeccionar acabados u ocultar imperfecciones con fines estéticos manteniendo siempre unas caracteristicas similares a las de la imagen original. Las herramientas más utilizadas en este tipo de manipulaciones suelen ser el saneado, perfilado, emborronado, difuminado y realce.

En la Figura 2.4 se han retocado las zonas correspondientes a la piel de la cantante Britney Spears con fines publicitarios para la portada de la revista 'Marie Claire'.

2.1.5. Huella Digital

Este tipo de manipulación no está centrado en la parte visual de la imagen si no en la información que ésta contiene.

La huella digital es un rastro que dejan todas las cámaras de los dispositivos móviles sobre la imagen que toman durante el proceso de captura. Cuando se genera una imagen digital se introduce este rastro, también llamado ruido.

(a) *Imagen original* (b) *Imagen manipulada*

Figura 2.3: Ejemplo de Manipulación mediante la aplicación de un filtro.

Figura 2.4: Ejemplo de Retoque Estético.

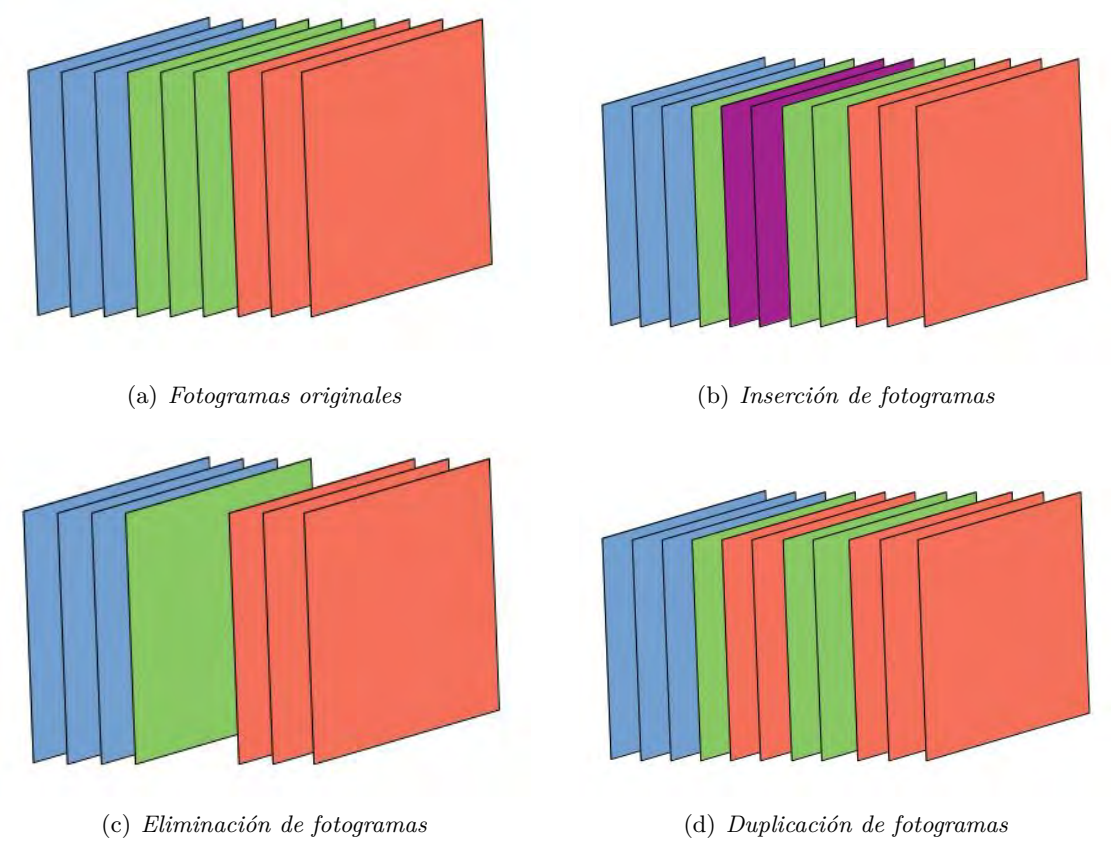
Extraer el ruido de una imagen proporciona una información valiosa acerca de la fuente (modelo y marca del dispositivo) que generó dicha imagen ya que el tipo de ruido que contiene pertenece sólo al modelo de la cámara que lo generó.

El objetivo de manipular la huella digital de una imagen es el de poder modificar su origen. Si se sustituye la huella digital de la imagen por otra, es posible incriminar a otro dispositivo móvil en la escena en cuestión. También es posible eliminar la huella y así anonimizarla.

2.2. Técnicas de Manipulación en Vídeos

2.2.1. Inter-Fotograma

Un vídeo digital es una secuencia de imágenes llamadas fotogramas, este tipo de técnica de manipulación se centra en la modificación de la correlación temporal entre ellos. Para modificar la correlación temporal del vídeo es posible insertar, duplicar, intercambiar o eliminar cualquiera de los fotogramas que lo conforman (Figura 2.5).



(a) *Fotogramas originales*

(b) *Inserción de fotogramas*

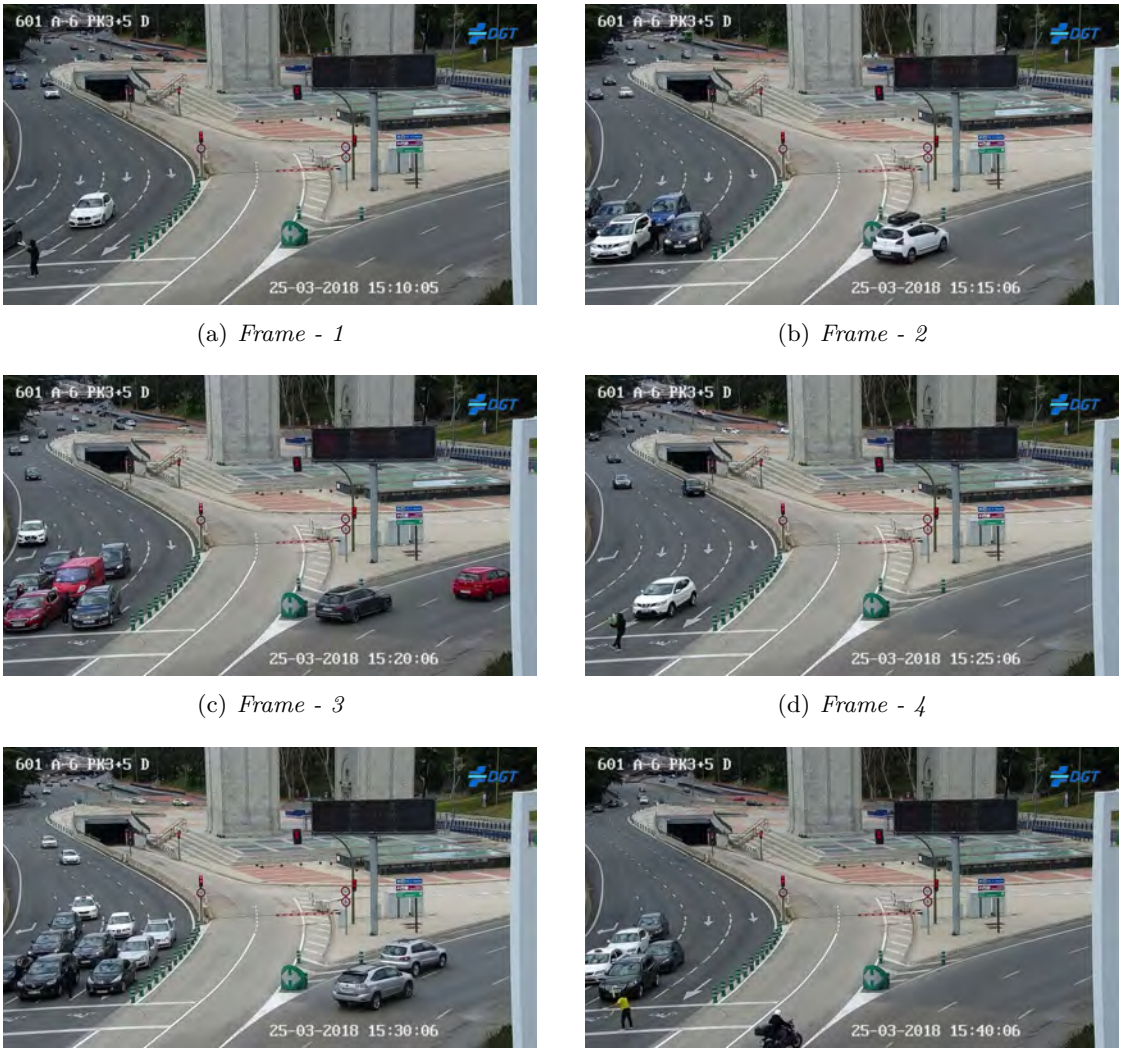
(c) *Eliminación de fotogramas*

(d) *Duplicación de fotogramas*

Figura 2.5: Ejemplo de Manipulación Inter-Fotograma

Otra forma de manipular un vídeo inter-fotograma es mediante el empalme de dos o más vídeos, es decir, interpolando fotogramas de ambos para generar uno nuevo. Además, es posible que los vídeos orginales no compartan los mismos fotogramas por segundo (fps) o frame-rate, por lo que será necesario también manipular esta caracteristica para ajustar los fps de uno al otro.

El principal objetivo de esta manipulación es el de eliminar de la escena grabada un evento indeseado. También es posible incriminar en la escena a otros objetos con la adición de un fotograma externo. Si se toma como ejemplo la secuencia de las imágenes de vigilancia de una cámara de tráfico, como las de la Figura 2.6. es sencillo hacer que el vehiculo blanco de la Figura 2.6(d) desaparezca de la escena eliminando ese fotograma.



(a) *Frame - 1*

(b) *Frame - 2*

(c) *Frame - 3*

(d) *Frame - 4*

(e) *Frame - 5*

(f) *Frame - 6*

Figura 2.6: Ejemplo de fotogramas de una cámara de vigilancia de la Dirección General de Tráfico (DGT)

En general, el ojo humano no puede detectar diferencias entre el video original y el video con manipulación inter-fotograma pero las operaciones de procesamiento de la manipulación dejan una huella en la información del contenido del vídeo.

2.2.2. Intra-Fotograma

La manipulación intra-fotograma se centra en la alteración de cada fotograma individualmente. Estas manipulaciones pueden clasificarse en:

- **Manipulación a nivel de pixel**: La cual consiste en tratar al fotograma como una imagen individual y aplicar técnicas de manipulación en imágenes como las vistas en la sección anterior, por ejemplo, copia-pega o empalmes.

- **Manipulación a nivel de fotograma**: Mediante la cual se cambia de tamaño o se recortan las extremidades de un fotograma con el objetivo de ocultar cierto contenido del video que se ubique en los bordes del fotograma. Por ejemplo marcas de Hora y lugar de grabación.

A diferencia del ejemplo expuesto con las técnicas inter-fotograma, si se tiene como objetivo ocultar el paso de un vehículo de la cámara de vigilancia de la Figura 2.6 con técnicas intra-fotograma, en lugar de eliminar el fotograma en el que aparece, se podría hacer desaparecer con técnicas de copia-pega o incluso se podría re-escalar el fotograma y recortar la zona en la que aparece.

2.3. Herramientas de Edición Multimedia

Las tablas que se muestran a continuación contienen una breve descripción de las herramientas más utilizadas para realizar manipulaciones sobre imágenes y vídeos digitales.

Tabla 2.1: Principales Herramientas de Edición de Imagen Digital

Herramienta	Público	Ergonomía	Funcionalidades
Photo Filtre	Principiante	Intuitiva y asistida	Filtros predefinidos
Photosop Elements	Amateur	Fácil de utilizar e intuitivo	Funcionalidades predefinidas
Magix X PhotoGraphic Designer5	Todos	Muy rápido e intuitivo	Alto nivel de detalle
Photosop CS3	Profesional	Requiere tiempo de adaptación	Alto nivel de detalle
GIMP	Profesional	Requiere tiempo de adaptación	Alto nivel de detalle

Tabla 2.2: Principales Herramientas de Edición de Vídeo

Herramienta	Público	Ergonomía	Funcionalidades
Video Easy	Principiante	Depurada al máximo	Simples y con asistente
Pinnacle Studio HD	Amateur	Manejo muy rápido	Funcionalidades predefinidas
Adobe Premiere CS6	Profesional	Requiere tiempo de adaptación	Alto nivel de detalle
Final Cut	Profesional	Requiere tiempo de adaptación	Alto nivel de detalle. Solo MAC
AVID6	Profesional	Requiere tiempo de adaptación	Alto nivel de detalle

Capítulo 3

Técnicas de Detección de Manipulación en Imágenes Digitales y Vídeos

En este capítulo se analiza el estado del Arte haciendo un recorrido por las principales técnicas de análisis forense en la detección de manipulaciones en imágenes y vídeos digitales.

3.1. Detección de Manipulación en Imágenes Digitales

La Figura 3.1 Muestra una clasificación de las técnicas de detección de manipulaciones en imágenes digitales

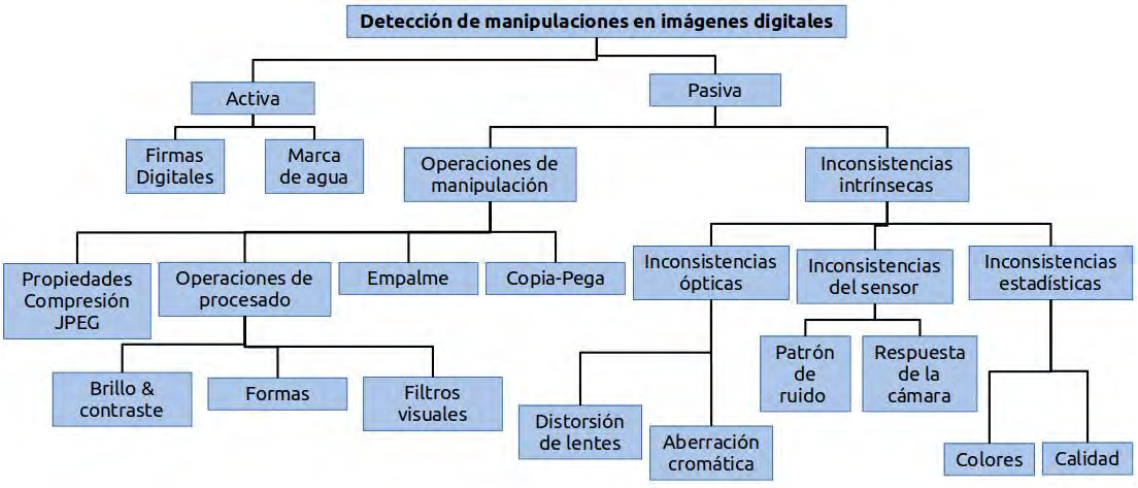


Figura 3.1: Esquema de Detecciones de Manipulación en Imágenes.

3.1.1. Detección de Copia-Pega

La detección de falsificaciones copia-pega son las técnicas más utilizadas en el campo forense debido a su simplicidad y eficacia. La principal evidencia que se explota para detectar este tipo de manipulación es la existencia de dos áreas iguales basándose en las propiedades de los bloques en los que se divide la imagen.

La primera aproximación que se realizó para identificar áreas copiadas fue realizada en el año 2003, en [FSL03] los autores propusieron un método que hacia uso Transformada Discreta del Coseno (DCT) para localizar coincidencias entre bloques de una manera más eficiente que la de realizar una búsqueda por fuerza bruta.

En [CPF04] se propone un método que utiliza el Análisis de Componentes Principales (PCA) para representar una imagen como una representación de bloques superpuestos. Obtuvieron unos resultados más eficientes que los obtenidos en la técnica anterior debido a que consiguieron reducir el coste computacional al rebajar a la mitad el número de cálculos requeridos para procesar los bloques con PCA. Aún así, el coste computacional seguiría siendo grande y por ello, en [LG06] se propone encontrar las coincidencias entre los bloques buscando patrones de intensidad similares.

Los autores de [LHQ06] proponen reducir aún más el tamaño de los bloques superpuestos en los que se subdivide la imagen para mejorar la precisión al comparar similitudes entre ellos. Las posibles áreas duplicadas tendrán unas propiedades de intensidad similares. Este método es más firme frente a pérdidas por compresión que los métodos anteriores.

En [WDT] se propone detectar las regiones duplicadas estudiando todas las invariantes de desenfoque de una imagen. Los resultados de éste método fueron correctos pero con la desventaja de obtener un tiempo de computación demasiado grande (un promedio de 30 minutos para una imagen RGB de tamaño medio).

En [MVP07] los autores desarrollaron un método que descomponía la imagen en registros de coordenadas polares y, haciendo uso de la Transformada Wavelet, detectar las regiones copiadas. Se reducía así la dimensionalidad de la imagen de entrada debido a la aplicación de Wavelet. Para encontrar los bloques similares se realizaba una búsqueda por fuerza bruta mapeando cada uno de los bloques con las coordenadas polares y la correlación entre ellos como criterio. Otros autores en [WWZ11] hicieron algo similar pero se basaron en la Transformada de Fourier (FFT).

En [LWTS07] aparte de utilizar la transformada Wavelet para reducir la dimensionalidad de la imagen, utilizan la Descomposición en Valores Singulares (SVD) (Singular-value decomposition) para generar el vector de características de cada región con el fin de buscar las similitudes con mayor eficacia. Las regiones duplicadas eran localizadas por clasificación lexicográfica y vecindad detectanto todos los bloques, incluso cuando la imagen había sido muy comprimida.

Los métodos descritos hasta ahora no producían resultados óptimos cuando las imagenes sufrían cierta transformación geométrica. En [HGZ08] los autores proponen una nueva metodología basada en el algortimo Scale-Invariant Feature Transform (SIFT) para estimar los parámetros de la transformación geométrica aplicada sobre la imagen (traslación horizontal o vertical, escalados o rotación del ángulo) con alta fiabilidad, pudiendo así, detectar falsificaciones en imágenes que han sufrido alguna de estas transformaciones. Los autores de [BJGY10] mejoran la robustez de SIFT proponiendo un método basado en el algoritmo Speeded Up Robust Features (SURF). Este método es además capaz de detectar también áreas copiadas a las que se le ha aplicado modificaciones en brillo o contraste.

3.1.2. Detección de Empalme

El empalme de imágenes es uno de los esquemas de manipulación más simples y comúnmente utilizados. La detección de este tipo de manipulación es una tarea fundamental durante la verificación de la integridad de imágenes. Por lo general, todas las técnicas se basan en las variaciones que se encuentran en el patrón de caracteristicas del área pegada respecto del contenido de la imagen original.

El primer método propuesto fue presentado en [Far99]. En este trabajo se propone una técnica basada en el análisis de la señal de la imagen para detectar las correlaciones no naturales que se introducen durante el proceso de falsificación. Obtuvo buenos resultados cuando la detección se llevaba a cabo sobre empalmes realizados por personas y no por máquinas.

En [NC04] los autores presentaron un modelo de detección de empalme de imágenes basado en el uso de características de magnitud y fase de la propia imagen. Los resultados de la precisión de detección fueron de aproximadamente del 70 %. Posteriormente, los mismos autores propusieron un método para detectar empalmes abruptos utilizando las mismas características.

Los autores de [FSS06], por su parte, se apoyaron en la transformada de Hilbert-Huang para generar estadísticas con el fin de utilizarlas para la clasificación de un modelo de imagen natural. Este modelo estaba basado en los momentos característicos obtenidos con ayuda de la descomposición de Wavelet y así conseguir distinguir las imágenes empalmadas de las imágenes auténticas.

Los autores de [HC06] propusieron extraer las invariantes de geometría de los píxeles de cada región de una imagen para estimar la Función de Respuesta de la Cámara (CRF) y estudiar las variaciones entre las distintas zonas de la imagen para, así, detectar las áreas que han sufrido un empalme.

En [SCC07] los autores investigaron las características estadísticas de los bloques de una imagen para detectar empalmes. Estas características son extraídas de matrices 2D, las cuales son generadas al aplicar al bloque de imágenes de varios tamaños la DCT. Los

experimientos tuvieron una precisión del 91 %.

En [ZKR08] se presentó un método de detección basado en caracteristicas de momento extraídas de la DCT y en métricas de calidad de imagen extraídas de la propia imagen. Descubrieron que ambas características sufrían variaciones cuando una imagen había sufrido un empalme y explotaron dichas variaciones.

Los autores de [LS09] sugieren un método basado en dividir la imagen por áreas para después extraer las características de densidad de los coeficientes DCT vecinos de cada área. Todas las variaciones en las densidades se clasifican mediante un SVM para identificar si esas áreas son diferentes.

En [ZCZ+09] se construye un método de detección de empalmes basado en el estudio de las sombras de la imagen. Mediante combinaciones y estimaciones de las zonas de sombra logran encontrar bloques empalmados. Los mismos autores en [ZCQ+10] utilizan la teoría de la homografía plana para localizar la región manipulada y aparte, desarrollaron un método de extracción automático que segmentaba el objeto falso de la imagen manipulada.

En [ZLLW10] los autores proponen un metodo basado en la comparación de los espacios cromáticos que forman la imagen. Se utilizan cuatro vectores RLRN con diferentes direcciones extraídos de los canales de crominancia que tienen correlación con características utilizadas en la detección de empalmes.

Los autores de [LCDG11] investigaron una técnica basada en el estudio de la iluminación de los objetos de una imagen. Se basan en la consistencia que debe existir en las sombras en función del grado de iluminación y en las características de color del valor de dicha sombra.

En [WIWS15] se evaluó la técnica ELA en imágenes manipuladas con distintos métodos, demostrando una cque sólo era efectiva para detectar empalmes.

En [GHK+17] se propuso un algoritmo basado en el uso de ELA que demostró detectar con éxito la imagen modificada y el punto exacto de la modificación mediante el uso de histogramas.

En [JBdSC17] utilizan la Transformada de Wavelet para filtrar los resultados de aplicar ELA con el objetivo de resaltar las alteraciones sobre la imagen.

3.1.3. Detección de Manipulación de Huella Digital

La huella digital de una imagen identifica su origen y garantiza su integridad. Las técnicas para detectarla se encargarán pues, de estudiar si dicha huella ha sufrido modificaciones, pues en tal caso, es una evidencia de que la imagen en cuestión ha sido manipulada. Las técnicas se basan principalmente en el estudio de los patrones del ruido del sensor que introduce cada cámara en las imágenes que genera durante el proceso de captura de una fotografía.

En [CAGSO+13] proponen un método basado en la extracción de características del Patrón de Ruido de Respuesta no Uniforme (PRNU) junto con un SVM para su

clasificación. Este trabajo se utilizó únicamente en dispositivos móviles y se consiguió mostrar que este método consigue buenos resultados cuando se tiene que clasificar una gran cantidad de cámaras fuente.

En [SOAGC+14] se propone combinar dos métodos de detección: el estudio de las imperfecciones del sensor y las Transformada Wavelet (WT). Los resultados confirman que estas dos técnicas juntas ayudan a rastrear con precisión el dispositvo fuente que tomó la imagen, además del modelo y marca de dicho dispositivo.

3.2. Detección de Manipulación en Vídeos

La Figura 3.2 Muestra una clasificación de las técnicas de detección de manipulaciones en vídeos digitales

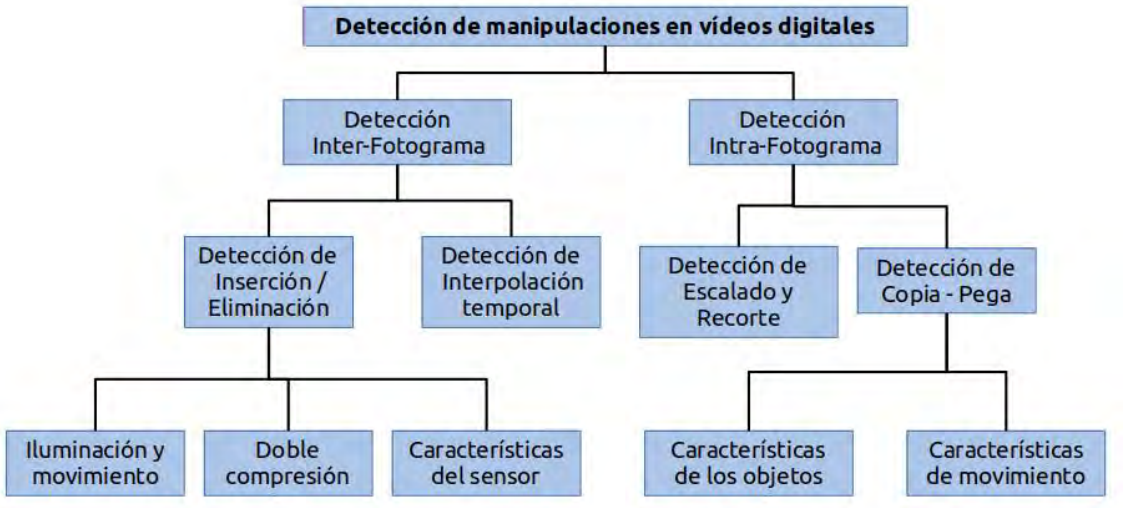


Figura 3.2: Esquema de Detecciones de Manipulación en Videos.

3.2.1. Detección de Manipulación Inter-Fotograma

3.2.1.1. Detección de Inserción / Eliminación / Duplicación de Fotograma

Los dispositivos introducen un ruido en cada fotograma cuando graban un vídeo. Dado que este ruido sigue un patrón particular en una secuencia de frames consecutivos, es posible que, investigando estos rastros, se detecten cambios entre los fotogramas.

En [DCG06] los autores utilizaron la varianza entre el ruido promedio de los frames y uno en particular. Los frames con varianzas más altas serían marcados como inserciones. No se demostró su eficacia sobre vídeos comprimidos, y además, su eficacia fue probada sobre vídeos autograbados y no es suficiente para determinar su aplicabilidad.

Los autores de [WF07a] propusieron un algoritmo de movimiento adaptativo que fue capaz de detectar y localizar falsificaciones en videos entrelazados y desentrelazados. Se

basaban en la detección de las perturbaciones en la correlación para los entrelazados y en los disturbios de movimiento entre frames para los desentrelazados. Este método, no obstante, resultaba ineficaz para vídeos de baja calidad.

En [MCP+07] utilizan el concepto del Patrón de Ruido del Sensor (SPN) de la cámara para determinar si todos los frames del vídeo habían sido grabados con el mismo dispositivo. Los resultados obtenidos indicaron que el algoritmo era fiable para vídeos no comprimidos, pero el rendimiento se deterioraba para vídeos comprimidos.

Otra propuesta basada en la cámara es presentada en [KOS10], que proporcionó autentificación a nivel de pixel para todos los frames del vídeo. Se basaba en las inconsistencias en los fotones del ruido de disparo que introduce la cámara durante el proceso de adquisición. También lograrían encontrar regiones sospechosas en los frames.

3.2.1.2. Detección de Interpolación Temporal entre Fotogramas

Otra forma de manipular un vídeo es mediante un corte temporal, es decir, intercalando frames de dos vídeos diferentes. Cuando se intercalan dos frames hay que tener en cuenta que es necesario sincronizar sus velocidades (frame-rate).

El método sugerido en [BBM+13] se basa en la propiedad de interpolación compensada por movimiento ya que deja huellas detectables en los frames. Los autores pudieron sugerir un sistema que funcionaba para videos no comprimidos y ligeramente comprimidos (por ejemplo, H.264, o videos de transmisión de televisión) y lograba resultados prometedores, incluso cuando se usaba sólo en un subconjunto de fotogramas. Además, el sistema funcionaba bien en ventanas espaciales de pequeño tamaño, lo que permitió que este detector se usara como una posible herramienta para detectar ataques de falsificación de copiar y pegar. Sin embargo, el número de cuadros interpolados observados tenía que ser lo suficientemente grande para que el sistema detectara las falsificaciones con éxito.

En [YYSL16] se detecta la conversión de velocidad ascedente de frames basándose en la intensidad de los bordes. Utilizan un umbral determinado para distinguir las zonas originales de las convertidas al alza, y en base a ello, estiman la velocidad teórica de los frames originales. Para un total de 300 secuencias de prueba, consiguieron un promedio de detección de 95 %.

Los autores en [XYL+17] desarrollaron un método de detección ciego basado en el análisis a nivel de frame de una característica llamada ”variación media de la textura”(ATV). Cada curva ATV generada se procesaba en el vídeo candidato como evidencia de la conversión de velocidad ascendente. Esta técnica podría localizar la posición de la interpolación de los frames y ayudar a estimar la velocidad original de los mismos.

3.2.2. Detección de Manipulación Intra-Fotograma

3.2.2.1. Detección de Copia-Pega

Las técnicas de detección de copia-pega proceden buscando similitudes entre regiones de frames sucesivos o dentro del mismo frame.

Para detectar estas falsificaciones, los autores en [WF07a] calcularon coeficientes de correlación espacial y temporal para identificar y localizar semejanza entre partes separadas del vídeo. Este método, obtuvo muy buenos resultados para vídeos con compresión MPEG debido a que los artefactos de compresión son más pronunciados en presencia de movimiento en el video.

Otra técnica propuesta en [Che10] y [GC11] se basó en la hipótesis de que los atributos de correlación de sub-bloques de píxeles intra e inter frame están obligados a ser desorganizados debido a alteraciones como doble compresión, retoque o remuestreo. Los autores extrajeron los residuos de ruido y la cuantificación de características de frames adyacentes para luego realizar un análisis de correlación usando el análisis de correlación canónico, análisis factorial intermodal, y análisis semántico latente. Tales perturbaciones ayudaron a la técnica para diferenciar las huellas de un video original de las de uno manipulado.

En [DDSD12], se propone detectar alteraciones con la conversión del video a una secuencia de frames, seguido de un proceso de emparejamiento de bloques dentro de la región sospechosa. Al trabajar en una parte del frame en lugar de todo el frame, la técnica es capaz de mantener un buen equilibrio entre rendimiento y complejidad.

El método de detección y localización de falsificación de [BMTT13] era similar en funcionalidad a [WF07b] pero de manera completamente automática. Es un algoritmo de dos pasos, en el que primero se detectan manipulaciones a nivel de frame y se analiza el video residual que se obtiene al restar píxeles que ocupan la misma posición espacial en frames consecutivos. Entonces, para detectar el contenido duplicado, los autores ponen en relación los bloques 3D de los frames. Así, la presencia de alta correlación indica la ubicación del contenido idéntico.

En [LT14], los autores propusieron un enfoque para detectar y localizar falsificaciones a nivel de región en videos. El método detecta irregularidades en la coherencia espacio-temporal entre frames consecutivos. El vídeo primero se divide en conjuntos de frames y luego se calcula la coherencia entre cada una de estos conjuntos. Dicho así, los conjuntos con coherencia antinaturalmente alta o coherencia anormalmente baja se clasificarían como frames manipulados.

Una técnica de localización y detección de eliminación de objetos es la que se nos presenta en [PSSA14]. Se utilizó aquí SIFT junto con la coincidencia k-NN y correlación cruzada ruido-residuo para detectar falsificaciones copia-pega. Aunque la técnica funciona bien para la prueba videos, esta sufre una degradación significativa a medida que aumenta la resistencia a la compresión.

3.2.2.2. Detección de Escalado o Recorte

Otra forma de manipular el contenido del vídeo es ampliando un frame y después eliminando el evento incriminatorio recortando la parte externa de éste. Es importante saber que cuando se hace un recorte se produce un remuestreo para mantener una resolución constante en todos los frames del vídeo.

En [HRLL13] los autores observaron que el remuestro introduce ciertas correlaciones estadísticas sobre el contenido dado. Explotaron el SPN como caracteristica forense y analizaron las variaciones en las propiedades de correlación de referencia SPN y el de re-escalado. Este método es bastate firme en cualquier tipo de vídeo, pero también resulta excesivamente dependiente de una gran cantidad de parámetros y umbrales dependientes del contenido, lo cual requiere un ajuste empírico extremadamente cuidadoso.

3.2.3. Detección de Doble Compresión

La recompresión o doble compresión es, una consecuencia inevitable de la falsificación, y su detección podría ayudar a detectar la presencia manipulaciones.

Los primeros pasos en esta dirección se pueden atribuir a los autores de [WF06]. Su algoritmo se basaba en la suposición simple de que cuando se manipulaba un video MPEG, se producían dos compresiones: primero, cuando se creaba el video y, segundo, cuando se volvían a guardar después de dicha alteración. También explotaron el hecho de que dentro de un Grupo de Imágenes (GOP), los frames muestran una gran correlación entre ellos, de manera que al agregar o eliminar un frame en un GOP aumenta el error de estimación de movimiento, lo que también da como resultado picos periódicos detectables.

En [WF09], los autores presentaron una técnica para detectar la cuantización doble, que resultó de la recompresión de un video comprimido MPEG o de la combinación de videos de características diferentes. La técnica podría detectar una manipulación si los coeficientes DCT de los frames del video se sometieron a doble compresión en cualquier punto. Los resultados empíricos indicaron que la tasa de detección fue altamente dependiente de la relación de la primera y la segunda escala de cuantificación. Evidentemente, la técnica fue efectiva siempre que el segundo factor de calidad de compresión fuera más alto que el primero.

En [SX10], las falsificaciones en videos codificados en MPEG-2 se detectaron mediante el examen de la distribución del coeficiente DCT. Este algoritmo se basó en la observación de que el histograma de coeficientes de DCT cuantificados de un video que había experimentado una doble compresión exhibía un patrón convexo. A diferencia de [WF09], que dependía en gran medida de las escalas de cuantificación, los autores en este caso sugirieron controlar la tasa de bits de salida, lo que hizo que este algoritmo se adaptara a las necesidades de diferentes tipos de sistemas de codificación de video pero no pudo localizar la falsificación en el video. Tampoco pudo funcionar bien para videos de cámara lenta.

El trabajo en [SNZ11] también se centró en la detección de alteraciones basadas en frames al detectar la compresión doble en videos MPEG-2. En lugar de basar el proceso de detección de agregación/eliminación de trama en las características temporales, los autores sugirieron utilizar las características de frecuencia. Se observó que cuando se vuelve a comprimir un video MPEG después de agregar/eliminar el frame, se pierden algunos componentes de alta frecuencia en los frames recomprimidos debido a la desincronización de los GOP y la cuantificación no lineal realizada en el proceso de codificación. Estas variaciones no solo ayudan a detectar la falsificación sino también a localizarla.

Otra técnica de detección de falsificación basada en doble compresión MPEG es la propuesta en [SWJ12], donde las anormalidades en los patrones de coeficientes DCT son tratadas como indicativo de inserción/eliminación de frames. Los autores extrajeron características de los GOP, que luego son utilizados por una SVM para determinar la velocidad de bits original del video doblemente comprimido dado, y se observa que el rendimiento de detección de la técnica era relativamente inferior para los videos con menor tasa de bits, porque una escala de cuantificación mayor requiere un proceso de cuantificación más robusto, que la técnica no estaba preparada para manejar.

En el mismo año, se pre-planteó una técnica similar en [XSY12] aunque con una novedad: su capacidad para detectar videos transcodificados, es decir, videos que habían sido doblemente comprimidos utilizando dos estándares de compresión diferentes. Los autores observaron además que después de que un video MPEG-2 se transformara en video MPEG-4, las trazas de compresión MEPG-2 anteriores, estos generan nuevas periodicidades que se observaron claramente en los histogramas de los coeficientes DCT reconstruidos. Los autores presentaron los resultados en forma de curvas de características operativas del receptor y declararon que se habían obtenido resultados perfectos en caso de bajas tasas de bits. Estas curvas también demostraron que a medida que aumentaba la velocidad de bits de salida objetivo, el rendimiento de detección disminuía. También asumió que la transcodificación siempre sugería manipulación.

Por otra parte, en [GFB+14] se propone detectar la codificación doble incluso si el conjunto de fotogramas principales hubiera sido eliminado. Este método tiene la ventaja adicional de poder ubicar efectivamente la falsificación, además de resultar adecuado también para videos codificados H.264, a diferencia de [XSY12] que funcionaba solo para videos MPEG. La metodología modificada también fue capaz de estimar el número de frames borrados.

Los autores en [WBI+14] declararon que las compresiones múltiples eran un tema poco explorado y que era arriesgado hacer suposiciones con respecto a la autenticidad del contenido digital simplemente sobre la base de la presencia de doble compresión. Su afirmación fue respaldada por el simple hecho de que el contenido digital disponible en Internet, generalmente, sufre más de una compresión.

En [JWS+13] utilizan las estadísticas de Markov para detectar doble compresión. Se basan en que la cuantización doble con diferentes parámetros inevitablemente introducirá

errores de redondeo, dejando artefactos detectables. El proceso aleatorio de Markov podría capturar dichos artefactos para la detección.

En [CJS+16] se basan en características estadísticas de los macrobloques de los P-frames. Proponen detectar la compresión doble de MPEG con el mismo QS. La extracción de características se produce durante la compresión repetida del video en el mismo factor de calidad.

En [JHS+18] analizan la degradación que se produce durante una recompresión encontrando que la variación de las características de un vídeo tienden a estabilizarse tras múltiples recompresiones.

En [AB18] estudian los efectos de la recompresión en los fotogramas predictivos para generar un vector de características con el cual detectar doble compresión a nivel GOP.

Para un video ordinario (descargado de Internet o grabado con ciertos dispositivos móviles), la presencia de signos de doble compresión puede no ser sospechosa pero tampoco debe considerarse inocua. Si se supone que un vídeo ha sido inalterado, la doble compresión no debería aparecer en dicho vídeo. Por otro lado, si un video dado muestra signos de doble compresión, indicaría la presencia de algún tipo de modificación no autorizada.

Por lo tanto, la presencia de signos de doble compresión serviría como primera, y posiblemente, más importante evidencia de alteración en vídeos.

Capítulo 4

Contribuciones

En este trabajo se implementa mediante dos métodos de detección forense diferentes. El primero de los métodos desarrollados (explicado a continuación) está enfocado en la detección de la existencia de la técnica de manipulación empalme en imágenes digitales basado en ELA. El otro método se centra en la detección de recompresiones en videos digitales, evidenciando así que el vídeo en cuestión ha sido manipulado. Para cada uno de estos métodos, primero se explican ciertos conceptos generales, esenciales para la compresión de los algoritmos desarrollados, y de esta manera, finalizar con la explicación detallada del algoritmo y su diagrama.

4.1. Detección de Empalme en Imágenes

4.1.1. Conceptos Generales

Para comprender bien el algoritmo que se explica en este apartado, es importante explicar previamente ciertos conceptos que contextualicen su utilización. Para la comprensión del algoritmo basado en la técnica de ELA es necesaria la explicación de la técnica en sí, así como las generalidades del formato JPEG y los modelos de color Red-Green-Blue (RGB) y Hue-Saturation-Value (HSV).

4.1.1.1. El Formato JPEG

JPEG es el método de compresión con pérdida más utilizado en imágenes digitales. El grado de compresión es ajustable, es decir, existe una compensación que puede variar a voluntad del usuario entre la calidad de la imagen y el espacio de almacenamiento que ocupa [Sny09].

- **La Transformada Discreta del Coseno**
  DCT es una variación de la transformada discreta de Fourier, donde la imagen se descompone en sumas de cosenos, pero utilizando únicamente números reales.

$$f_j = \frac{1}{2}(x_0 + (-1)^j x_{n-1}) + \sum_{k=1}^{n-2} x_k \cos\left[\frac{\pi}{n-1} kj\right] \tag{4.1}$$

DCT es utilizada por el algoritmo de compresión JPEG por su gran capacidad de compactar la información en un número reducido de coeficientes y por ser independiente del número de datos de entrada que recibe, garantizando así una mayor eficiencia al trabajar con imágenes de grandes dimensiones. La compresión de imágenes se lleva a cabo descartando las partes imperceptibles para el ojo humano. Este proceso usa los coeficientes DCT para diferenciar que puntos de la imagen presentan características diferentes al resto o cuales son similares, por lo que es muy usado en la detección de manipulaciones en imágenes: trabajar con DCT en el canal de crominancia va a permitir obtener los coeficientes que indicarán puntos de la imagen que sobresalen del resto pero que a simple vista no son perceptibles.

El método de compresión JPEG generalmente tiene pérdidas, lo que significa que se pierde parte de la información de la imagen original y no se puede restaurar, lo que posiblemente afecte la calidad de la imagen. Las pérdidas son acumulativas, es decir, frente a recompresiones la imagen se degrada aún más. También hay un modo opcional sin pérdidas definido en el estándar JPEG, sin embargo, este modo no es ampliamente compatible con la totalidad productos existentes del mercado.

4.1.1.2. Modelos de Color

Un modelo de color es un modelo matemático abstracto que proporciona un método para definir colores mediante componentes de color especificos. En otras palabras, un modelo de color establece el conjunto de colores primarios a partir de los que, mediante mezclas se pueden obtener otros colores hasta cubrir todo el espectro visible [elab]. En la Figura 4.1 se observan los modelos de color RGB y HSV.

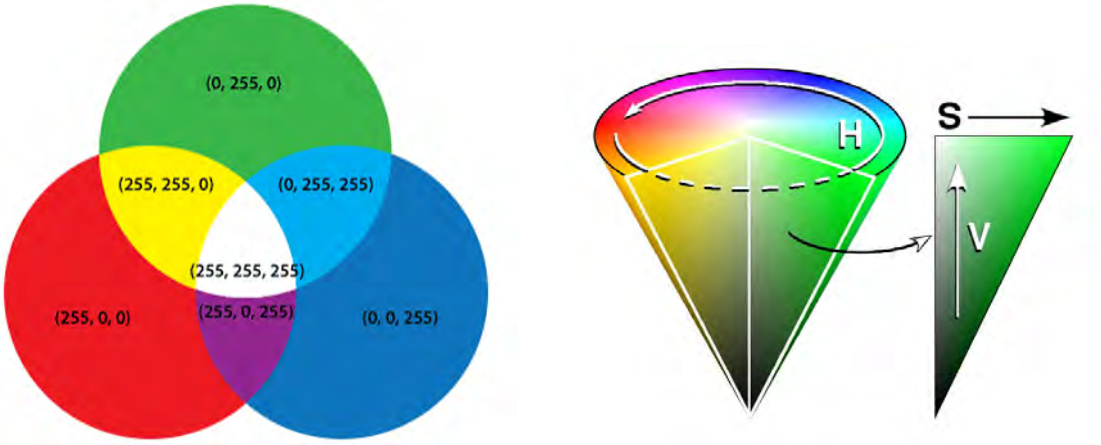
- **El Modelo RGB**:

  Utiliza los componentes rojo (R, del inglés Red"), verde (G, del inglés "Green") y azul (B, del inglés "Blue") para definir el grado de intensidad de cada uno de ellos en la representación de un color determinado. Cada una de estas componentes puede tener un valor comprendido entre el 0 y el 255. Este es lo que podemos llamar un modelo de color aditivo, es decir, el color se produce con la suma de las tres intensidades. El 0 indica ausencia de luz, es decir, el ojo percibe el negro. Por el contrario, con el 255 se percibe el blanco.

- **El Modelo HSV**:

  Utiliza el matiz (H, del inglés "Hue"), la saturación (S, del inglés "Saturation") y el brillo o valor (V, del inglés "Value") para definir colores. Es un sistema de

coordenadas cilindricas. Así el matiz expresa el color primario y se representa como el valor del ángulo de la circunferencia del cilindro, que puede tomar valores desde 0 a 360: el color rojo corresponde a 0º, el verde a 120º y el azul a 240º. La saturación tomará valores comprendidos entre 0 y 100 debido a que es el porcentaje de distancia al eje y ofrece diferentes valores que van desde el color propiamente seleccionado hasta el blanco. Por último, el brillo o valor corresponde con el porcentaje de altura en el eje blanco-negro: De esta manera, 0 corresponde al valor negro y 100, el blanco o el color con el nivel de saturación seleccionado.



(a) *RGB* (b) *HSV*

Figura 4.1: Modelos de Color

4.1.1.3. La técnica ELA

La técnica ELA se centra en la identificación de áreas con diferente nivel de compresión dentro de una misma imagen. Una imagen comprimida en formato JPEG debe tener aproximadamente el mismo nivel en todo su contenido. Si existe alguna zona con un nivel de error significativamente diferente, entonces, tiene una alta probablidad de que ha existido una manipulación digital sobre ella.

Existen en este sentido, varios formatos de imagen diferentes:

- **Formatos sin pérdida**: Los formatos sin pérdida conservan la información de color de píxeles exacta, es decir, si la imagen sufre recompresiones los pixeles de nuevo tendrán el mismo valor, incluso si se produce una conversión a otro formato sin pérdida. Por este motivo, estos formatos no son los adecuados para explotar este algoritmo. Los formatos sin pérdida principales son Portable Network Graphics (PNG) y Mapa de Bits (BMP).

- **Formatos con pérdida**: Los formatos con pérdida no garantizan que los colores permanezcan iguales. JPEG es un formato con pérdida, por ejemplo. Cuando se

comprime en este formato hay que establecer un nivel de calidad, el cual ajusta la cantidad de compresión que se aplica, es decir, la cantidad de información de color que se pierde. Por este motivo, guardar una imagen con JPEG hace que sus colores cambien ligera o imperceptiblemente dado que visualmente la imagen será igual, pero el valor de los píxeles habrá cambiado. Cuando la imagen se guarda por primera vez con JPEG se produce una gran cantidad de pérdida de color, sin embargo, para las siguientes recompresiones provocan una degradación de color significativamente menor.

Podría decirse que ELA resalta las áreas de la imagen más propensas a degradar sus colores en las próximas recompresiones debido a que las zonas editadas tienen un mayor potencial de degradación en comparación con el resto de la imagen. Concretamente, el algoritmo JPEG opera en una matriz de 8x8 píxeles, y cada cuadrado de 8x8 se comprime de forma independiente. Si la imagen no se modifica por completo, todos los cuadrados de 8x8 deberían tener potenciales de error similares, o lo que es lo mismo, que al recomprimirse cada cuadrado se degradará aproximadamente a la misma velocidad. ELA recomprime la imagen en un nivel de calidad JPEG específico. Este reguardado, por tanto, introduce una cantidad conocida de error en toda la imagen que se compara con la imagen original. Si se modifica la imagen, cada cuadrado de 8x8 afectado debería tener un potencial de error mayor que el resto de la imagen, por lo que las áreas modificadas aparecerán con un mayor nivel de error potencial.

A parte de conocer en que está basada ésta técnica es importante saber cómo interpretar el resultado. Principalmente podemos fijarnos en [elac]:

- **Bordes**: Si la imagen ha sido modificada, en el resultado ELA deben verse todos los bordes de alto contraste con brillos similares entre sí, lo mismo ocurre para todos los bordes de bajo contraste. En cambio, en una imagen original, los bordes de bajo contraste deberían ser casi tan brillantes como los bordes de alto contraste.

- **Texturas**: Las texturas similares deben tener una coloración similar en ELA. Las áreas con más detalles, como un primer plano, probablemente tendrán un resultado de ELA más resaltado que una superficie lisa.

- **Superficies**: Independientemente del color real de la superficie, todas las superficies planas deben tener aproximadamente la misma coloración en ELA.

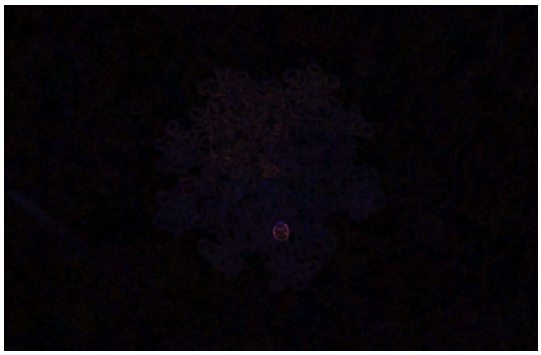
- **Calidad**: Al volver a guardar un archivo JPEG se eliminan las altas frecuencias y se obtienen menos diferencias entre los bordes, texturas y superficies de alto contraste. Una imagen JPEG de muy baja calidad aparecerá muy oscura. De la misma manera, escalar una imagen más pequeña puede aumentar los bordes de alto contraste, haciéndolos más brillantes bajo ELA. Del mismo modo, al guardar un archivo JPEG con un producto de Adobe (Photoshop) los bordes y las texturas de

alto contraste se agudizarán automáticamente, lo que los hará parecer mucho más brillantes que las superficies de baja textura.

Para obtener resultados óptimos con esta técnica es recomendable utilizar la imagen del punto de origen, ya que es la de mejor calidad, y es que si se utiliza una imagen que provenga de Internet es probable que esta ya haya sido comprimida.

Como otro ejemplo, debemos destacar los archivos PNG, un formato de archivo sin pérdida. Si una imagen es un PNG original, ELA debería producir valores muy brillantes para bordes y texturas. Sin embargo, si ELA genera resultados débiles, entonces el archivo PNG probablemente se creó a partir de un archivo JPEG. En resumen, la técnica Error Level Analysis (ELA) consiste en realizar un guardado intencionado de la imagen comprimiéndolo en una tasa de error JPEG conocida, con el fin de resaltar aquellas celdas de la imagen que han sido manipuladas mediante el cálculo de la diferencia entre ésta y la original [elaa]. En las Figuras 4.2, 4.4 y 4.4 se muestran ejemplos de la aplicación de esta técnica.

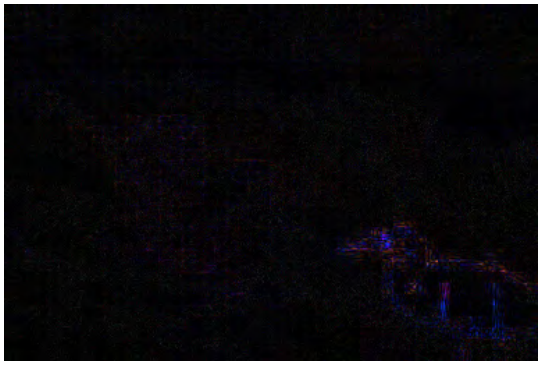
(a) *Una Imagen Manipulada con empalme*



(b) *Resultado ELA de (a)*

Figura 4.2: Ejemplo 1 de aplicación de la técnica ELA.

(a) *Una Imagen Manipulada con empalme*



(b) *Resultado ELA de (a)*

Figura 4.3: Ejemplo 2 de aplicación de la técnica ELA.

(a) *Imagen Original*

(b) *Resultado ELA de (a)*

(c) *Imagen (a) Manipulada con empalme*

(d) *Resultado ELA de (c)*

Figura 4.4: Ejemplo 3 de aplicación de la técnica ELA.

4.1.2. Algoritmo de Detección de empalme Basado en la Técnica ELA

Este algoritmo tiene el objetivo de detectar las zonas de la imagen que no pertenecen al contenido original. Está desarrollado en Python 2.7, utiliza librerias especializadas en proceso de imágenes como OpenCV y PIL. La entrada y salida del programa es una imagen.

Según la técnica ELA explicada en el punto anterior, una imagen JPEG original debería tener el mismo nivel de compresión a lo largo de todo su contenido. Cuando la imagen contiene una región que no pertenece a su contenido original los bordes y texturas de dicha zona quedarán resaltados del resto. Además, teniendo en cuenta que la compresión JPEG es ajustable podemos saber el nivel de compresión del contenido de la imagen.

- Entrada: Imagen o directorio de imágenes a analizar y el nivel de compresión JPEG.
- Salida: Imagen con la región que no pertenece al contenido original resaltada.

El algoritmo necesita dos entradas. La primera entrada al programa es un directorio que contenga una o varias imágenes JPEG. Para unos resultados óptimos sería deseable que estas imágenes tuvieran la máxima resolución posible y un número entero entre 0 y 100 que representa el nivel de compresión JPEG que queremos utilizar. Lo más recomendable es entre 85 y 95.

Por cada imagen se lanza un thread que se encargará de tratar dicha imagen y generar su salida. El primer paso para analizar la imagen es recomprimirla en formato JPEG con el nivel de calidad declarado como parámetro de entrada.

Una vez se tiene la imagen de entrada y su recompresión, se debe obtener la diferencia pixel a pixel y en valor absoluto entre ambas imágenes.

Con este resultado se puede identificar que zona de píxeles han sufrido un mayor cambio al aplicar el nivel de compresión recuperando los valores máximos y mínimos obtenidos.

Los valores de los píxeles que componen la imagen de salida del programa se calculan en función de los valores máximos y mínimos calculados previamente. Para ello, se escalan en base al valor 255.0 (RGB) y se potencia el brillo de cada uno.

Finalmente, se aplica una máscara sobre la imagen generada para resaltar todas las zonas que han quedado con tonos azules y rojos con más brillo que el resto del contenido. Como la máscara cubre las zonas menos brillantes se convierte la imagen RGB al modelo de color HSV. El objetivo de esta conversión es porque trabajar con valores HSV facilita la tarea de aislar colores. En la representación HSV del color, el tono determina el color que desea, la saturación determina qué tan intenso es el color y el valor determina la claridad de la imagen. Para aislar los colores, se deben aplicar máscaras múltiples. Una máscara de umbral bajo y una máscara de umbral alto para matiz, saturación y valor. Cualquier píxel dentro de estos umbrales se establecerá en 1 y los píxeles restantes serán cero. Estos umbrales son cofigurables a nivel de código. La conversión de RGB a HSV se rige por las siguientes fórmulas:

$$R' = \frac{R}{255.0}, G' = \frac{G}{255.0}, B' = \frac{B}{255.0}$$

$$C_{max} = max(R', G', B'), C_{min} = min(R', G', B')$$

$$\delta = C_{max} - C_{min}$$

$$H = \begin{cases} 60 \circ \frac{G'-B'}{\delta} mod 6 & C_{max} = R' \\ 60 \circ \frac{B'-R'}{\delta} + 2 & C_{max} = G' \\ 60 \circ \frac{R'-G'}{\delta} + 4 & C_{max} = B' \end{cases} \tag{4.2}$$

$$S = \begin{cases} 0 & C_{max} = 0 \\ \frac{\delta}{C_{max}} & C_{max} \neq 0 \end{cases} \tag{4.3}$$

$$V = C_{max}$$

Una vez se ha aplicado la máscara, la imagen de salida del programa queda con las zonas afectadas por empalme marcadas de un color blanco que resalta sobre el resto del contenido debido a que, o es una zona de color negro, o son zonas con píxeles de color blanco pero están aislados.

La imagen queda guardada en el directorio de salida para que el investigador pueda comprobar esas zonas que más resaltan y compararlas con la imagen de entrada para verificar si esa zona pertenece o no al contenido original.

El Diagrama de este algoritmo se encuentra en la Figura 4.5.

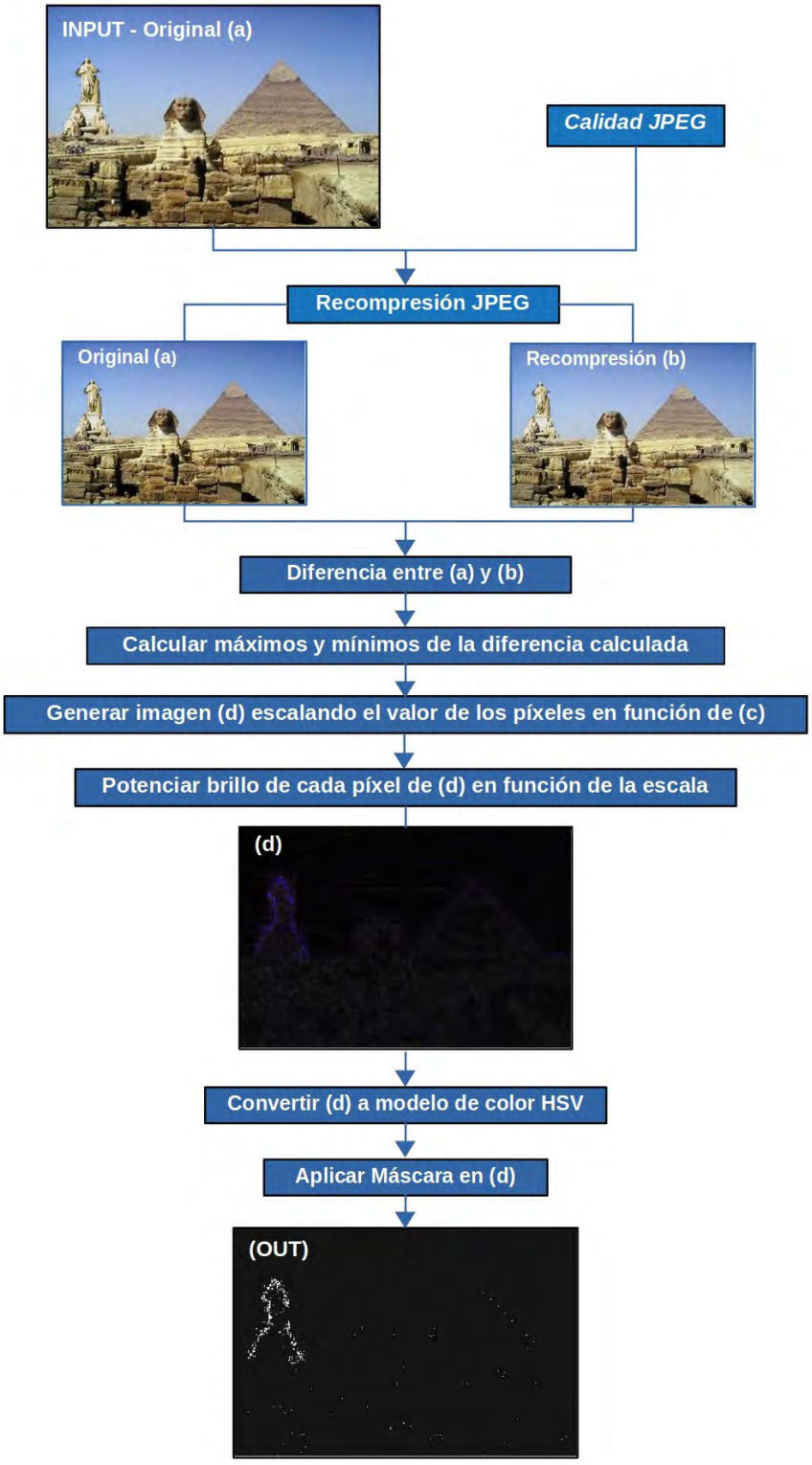


Figura 4.5: Diagrama del Algoritmo de Detección de empalme.

4.2. Detección de Doble Compresión en Vídeos

4.2.1. Conceptos Generales

Para una correcta comprensión de la explicación del algoritmo de detección de doble compresión primero es importante conocer las características del formato de vídeo H.264 y de las herramientas FFMPEG y LIBSVM, utilizadas ambas por dicho algoritmo.

4.2.1.1. El Formato H.264/MPEG4

Este estándar de codificación fue desarrollado con el objetivo de mejorar la calidad de la imagen, mejorar la eficiencia de codificación y mejorar la robustez de errores en comparación con normas anteriores como MPEG-2, H.263, etc...

El diseño de codificación de este estándar está basado en bloques, es decir, cada frame codificado se representa como una unidad de bloques llamados macrobloques. El algoritmo de codificación es el conjunto que se forma al predecir frames por medio de esos macrobloques para explotar dependencias estadísticas temporales, y al transformar la predicción residual para explotar las dependencias estadísticas espaciales.

Algunas de las características más destacadas del diseño, y que además permiten una mayor eficacia de codificación, incluyen las siguientes mejoras en la capacidad para predecir los valores del contenido del frame que se va a codificar:

- Por una parte, el tamaño de bloque de compensación de movimiento variable: Este estándar admite más felixibilidad en la selección de tamaños y formas de los bloques, con un tamaño de bloque mínimo de 4:4.

- Por otra, una referencia múltiple para la compensación de movimiento de un frame: Los frames con codificación predictiva, llamados P-frames, en estándares anteriores usan sólo el frame previo para predecir los valores del frame entrante. Este modelo extiende la codificación eficiente al permitir que un codificador seleccione, para fines de compensación de movimiento, entre un mayor número de frames que se han decodificado y almacenado en el decodificador.

El ojo humano percibe el contenido de una escena en términos de información de brillo y color por separado, y con mayor sensibilidad a la de brillo que la de color. El formato H.264 separa una representación de color en tres componentes llamados Y, Cb y Cr. El componente "Yrepresenta el brillo, mientras que los dos componentes de color Çbz Çrrepresentan la medida en la que el color se desvía del gris hacia azul y rojo, respectivamente. Debido a que el sistema visual humano es más sensible al brillo que al color, H.264 utiliza una estructura de muestreo en la que el componente cromático tiene un cuarto del número de muestras que el componente luminámico.

Cada frame se divide en macrobloques de tamaño fijo que cubren un área rectángular de 16:16 muestras de la componente de brillo y 8:8 muestras de cada uno de los dos

componentes de color. Los macrobloques son los componentes básicos para el que se especifica el proceso de decodificación. Todas las muestras de luminancia y croma de un macrobloque se predicen espacial y temporalmente. La señal de video de entrada se divide en macrobloques, cuya asociación se realiza en base a los tipos de frames a los que pertenecen, y luego se procesa cada macrobloque de cada tipo. Es posible un procesamiento en paralelo eficiente. [WSBL03]

En la compresion .H264 se puede seleccionar la predicción de los macrobloques de manera individual, en lugar de ser los mismos para todo el frame, de la siguiente manera: (ver Figura 4.6).

- **Frames - I**: Todos los macrobloques del frame son codificados usando intra-predicción, es decir, no utiliza la información codificada de otros frames.

- **Frames - P**: Ademas de la codificación de intra-predicción. También se pueden codificar usando inter-predicción con como máximo una señal de predicción de compensación de movimiento por bloque de predicción, es decir, su información proviene del frame previo.

- **Frames - B**: Además de los tipos de codificación disponibles en un frame P, algunos macrobloques del frame B también se pueden codificar utilizando la inter-predicción con dos señales de predicción de coimpensación de movimiento por bloque de predicción, es decir, su información proviene del frame previo y del siguiente.

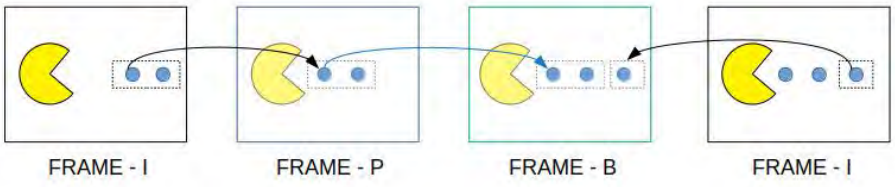


Figura 4.6: Secuencia de predicción de frame.

La Predicción entre frames P: La señal de predicción para cada macrobloque de NxM codificado predictivamente se obtiene desplazando un área de la imagen de referencia correspondiente, que se especifica mediante un vector de movimiento de traslación y un índice de referencia de imagen. Los componentes del vector de movimiento se codifican de forma diferencial usando predicción mediana o direccional de bloques vecinos. Ninguna predicción del componente del vector de movimiento (o cualquier otra forma de predicción) tiene lugar a lo largo de los límites del frame. En la Figura 4.7 se muestra un ejemplo de cómo actúa el vector de movimiento en la predicción. [WSBL03]

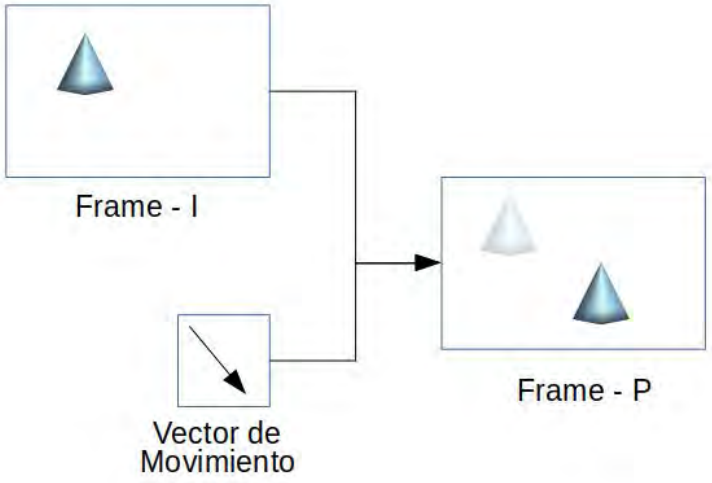


Figura 4.7: Secuencia de predicción de frame.

4.2.1.2. La Herramienta FFMPEG

FFMPEG es una plataforma de software libre multimedia capaz de decodificar, codificar, transcodificar, transmitir, filtrar y reproducir la mayoría de formatos de audio y vídeo. Está desarrollado en GNU/Linux pero también compila y ejecuta en la mayoría de sistemas operativos, entornos de desarrollo, arquitecturas y configuraciones. Tiene una licencia GNU LGPL, la cual garantiza una cierta libertad a la hora de compartir y modificar el software, asegurando que el software es libre para todos sus usuarios. [FFMb]

Componentes:

- **ffmpeg**: Línea de comandos para realizar cualquiera de las operaciones posibles.
- **ffserver**: Servidor de streaming multimedia de emisiones en directo.
- **ffplay**: Reproductor multimedia.
- **libavcodec**: Biblioteca que contiene todos los códecs de FFmpeg.
- **libavformat**: Biblioteca que contiene los multiplexadores/demultiplexadores para los archivos contenedores multimedia.
- **libavutil**: Biblioteca de apoyo que contiene todas las rutinas comunes.
- **libpostproc**:Biblioteca de funciones de postproceso de vídeo.
- **libswscale**: Biblioteca de escalado de vídeo.

Es posible utilizar FFMPEG para analizar los macrobloques y vectores de movimiento de cualquier archivo de vídeo MP4. En la Figura 4.8 se puede ver un ejemplo de un frame con los vectores de movimiento analizados impresos en forma de flechas. [FFMa]

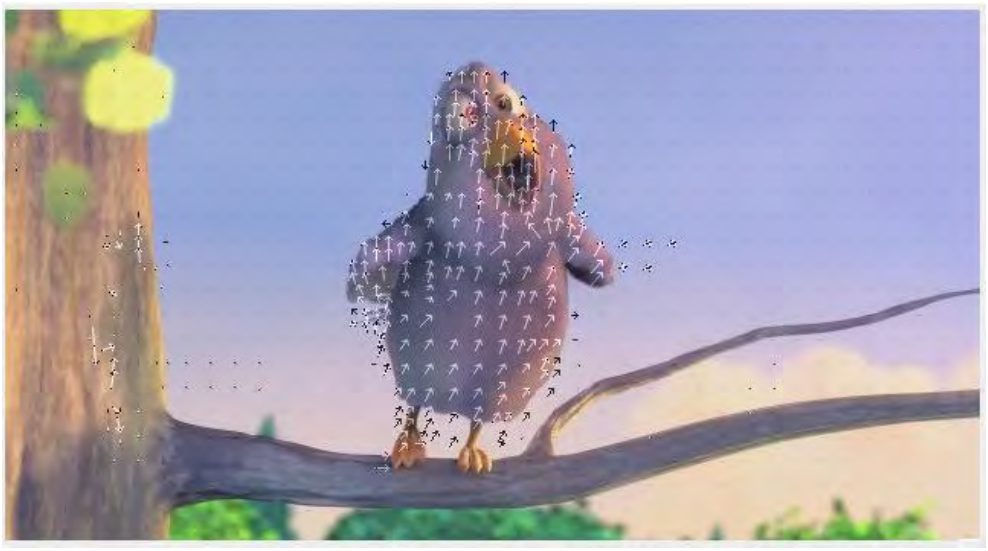


Figura 4.8: Análisis de los vectores de movimiento.

4.2.1.3. La Máquina de Soporte Vectorial

Las SVM son técnicas supervisadas de aprendizaje automático, muy útiles para la resolución de problemas de reconocimiento de patrones y para el análisis de regresión.

A partir de un conjunto de muestras la SVM construye un modelo que se utiliza para predecir la clase a la que pertenece una nueva muestra. El objetivo de la SVM es encontrar el mejor hiperplano que divida los datos de todas las muestras del entrenamiento en dos o más clases bien diferenciadas, es decir, determinar el hiperplano con la máxima distancia con el punto de cada clase que esta más cercano a éste.

En la Figura 4.9 el hiperplano H2 sería óptimo.

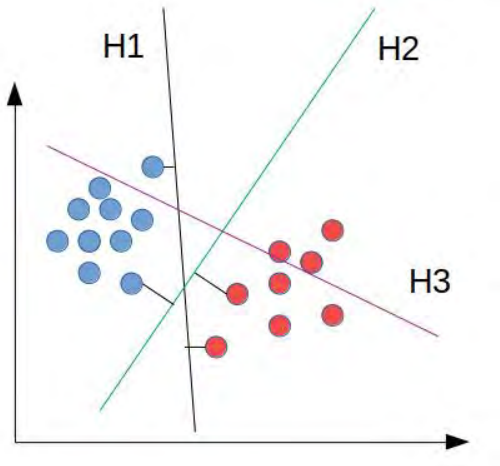


Figura 4.9: Muestras e hiperplanos.

No obstante, cuando se utiliza SVM surgen dos problemáticas:

- Los universos que se estudian utilizan más de dos dimensiones y no tienen una representación lineal. Este problema se soluciona con la representacion por funciones kernel, que proyectan la información a un espacio de características multidimensional mediante un mapeo no lineal.
- Seleccionar los parámetros apropiados del kernel. Hay dos parámetros en la función Función de Base Radial (RBF) del kernel (C y γ). Para encontrar los mejores parámetros de clasificación de prueba y entrenamiento se utiliza el método de optimización de parámetros. .

.

4.2.2. Algoritmo de Detección de Doble Compresión en Vídeos

Este algoritmo se utilizará con el fin forense de determinar si un vídeo ha sufrido más de una compresión, evidencia primera de que ese vídeo haya podido sufrir cualquier tipo de manipulación.

La detección de recompresiones está basada en el estudio de las características estadísticas del MBM [CJS+16].

El MBM es una característica que consta del tipo de macrobloque y vector de movimiento. Para extraer esta característica, un vídeo es recomprimido repetidamente en la misma escala de calidad para luego calcular el número de MBM diferentes entre dos compresiones secuenciales. Finalmente, estas estadísticas extraídas son utilizadas por la SVM para determinar si el vídeo es original o si ha sido recomprimido.

Este método viene inspirado de la convergencia de los coeficientes JPEG cuando se recomprime una imagen. Ambos métodos para cada recompresión varían en la forma de la Figura 4.10.

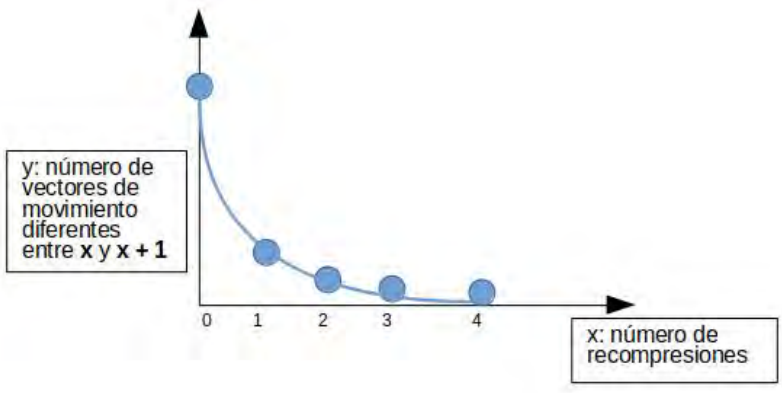


Figura 4.10: Número de MBM diferentes entre recompresiones.

A parte de los 3 tipos de frames (I, P, B) que tiene el estándar MPEG4, también existen 3 tipos de macrobloques:

- I-MB: Macrobloques con intra-codificación.
- P-MB: Macrobloques con inter-codificación.
- S-MB: Macrobloques saltados.

Un MBM está compuesto de las dos propiedades de la siguiente manera:

$$[MBM(M) = M_{type}, M_{mv}]$$

$$\left.\begin{aligned} M_{type} \in \{I-MB, P-MB, S-MB\} \\ M_{mv} = \{(u,v)|u, vcZ\} \end{aligned}\right\}$$

M es el macrobloque, M_{TYPE} el tipo del macrobloque M, y M-MV el vector de movimiento del macrobloque M.

Dos macrobloques se consideran que tienen el mismo MBM si y sólo si tienen el mismo M-TYPE y M-MV. Hay que tener en cuenta que cuando el M-TYPE es un I-MB su vector de movimiento es {0,0}, es decir, sólo es necesario evaluar los MBM diferentes de los P-frames y por tanto sólo hay que comparar el Vector de Movimiento (VM).

Para una secuencia de compresiones sobre un vídeo, si el macrobloque del mismo frame y misma posición de la compresión (n) y de la compresión (n + 1) tienen la característica MBM igual, se considera que ese macrobloque es estable. De otro modo se considera inestable. Ver Figura 4.11.

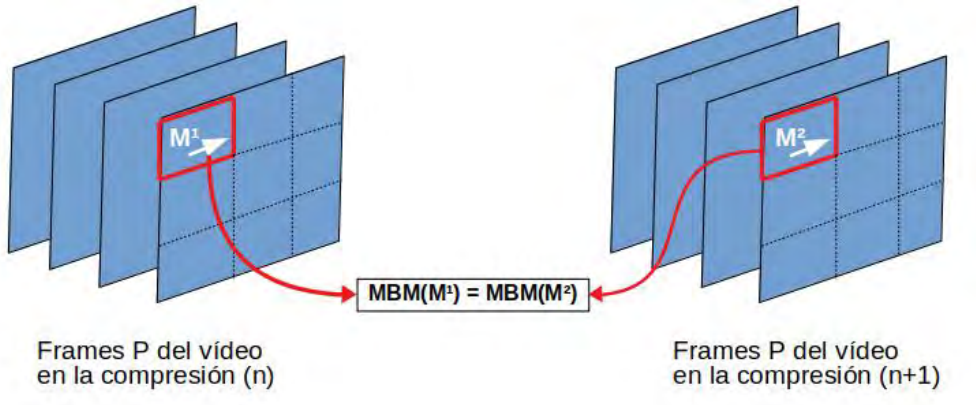


Figura 4.11: MBM estable.

A continuación se explica más detalladamente el algoritmo:

- Entrada: Vídeo o Vídeos en formato H.264-MP4
- Salida: Vector de características extraídas de los vídeos

Para un vídeo dado, este algoritmo devuelve un vector de características. En este caso el algoritmo se ha desarrollado para extraer tres características con el objetivo de que

la SVM las pueda clasificar hasta la triple compresión. También es posible extraer sólo dos y así discernir sólo entre vídeos originales y doblemente comprimidos, o incluso, para detectar más allá de la triple compresión, aunque, a partir de ésta, el rate de confianza de la máquina de soporte verctorial es demasiado bajo.

Vector de características:

- Número promedio de macrobloques inestables por P-frame encontrados entre el vídeo de entrada y su recompresión.
- Número promedio de macrobloques inestables por P-frame encontrados entre el vídeo recomprimido y su re-recompresión.
- Número promedio de macrobloques inestables por P-frame encontrados entre el vídeo de re-recomprimido y su re-re-recompresión.

En primer lugar, se evalua el MBM de los macrobloques. Para ello, con apoyo de la herramienta FFMPEG, se extraen los VM de los P-frames. Es importante tener en cuenta que para poder extraer información el vídeo debe estar primero en un formato crudo (.YUV). Estos vectores de movimiento contienen la información del frame al que pertenecen, la posición del macrobloque y la posición del eje X, Y de origen y destino del vector. Todos los vectores son almacenados en un archivo de texto para su posterior procesamiento.

Una vez realizado el proceso de extracción de VM, se debe recomprimir el vídeo de entrada con el formato H264-MP4. Esta recompresión tiene que tener la misma escala de calidad que el original, es decir la recompresión se realiza utilizando las mismas características del vídeo de entrada (qs, ancho, alto, rate, etc).

El vídeo recomprimido es almacenado para volver a ejecutar los pasos anteriores: Extracción de vectores de movimiento y, a partir de ahí, una nueva recompresión. Estas acciones pueden realizarse el número de veces que se quiera según el nivel de precisión de detección que se pretenda alcanzar. Para este caso se han realizado hasta tres recompresiones.

Cuando se alcance el número de recompresiones indicado, se va a disponer de tres archivos de texto que contienen los vectores de movimiento del video de entrada y de sus recompresiones posteriores. Con toda esta información se puede proceder a calcular el número promedio de macrobloques inestables por P-frame (C).

La siguiente fórmula muestra cómo se realiza ese cálculo:

$$C_n = \frac{1}{N} \sum_{i,x,y} I(M_n(i,x,y), M_{n+1}(i,x,y))$$

N es el número total de P-frames y M muestra el macrbloque de la recompresión enésima localizado en (x, y) del P-frame iésimo.

I se define como:

$$\left.\begin{array}{l} 1 \rightarrow MBM(M_1) \neq MBM(M_2) \\ 0 \rightarrow MBM(M_1) = MBM(M_2) \end{array}\right\} \quad I(M_1, M_2)$$

Para realizar el cálculo de M(i,x,y), los vectores de movimiento contenidos en los ficheros de texto son tratados en forma de matrices NxM dónde N es VM y M el P-frame al que pertenece con el fin de facilitar la tarea de la comparación.

Una vez encontrado I, es decir, el número de MBM diferentes de los ficheros de texto correspondientes a la compresión (n) y a la compresión (n + 1), se divide entre el número de P-frames.

El resultado se almacena en la posición (n) del vector de características del vídeo en cuestión.

Cuando el vector de características está completo se formatea para que la máquina de soporte vectorial lo pueda utilizar para realizar las tareas de clasificación.

El Diagrama de este algoritmo se encuentra en la Figura 4.12.

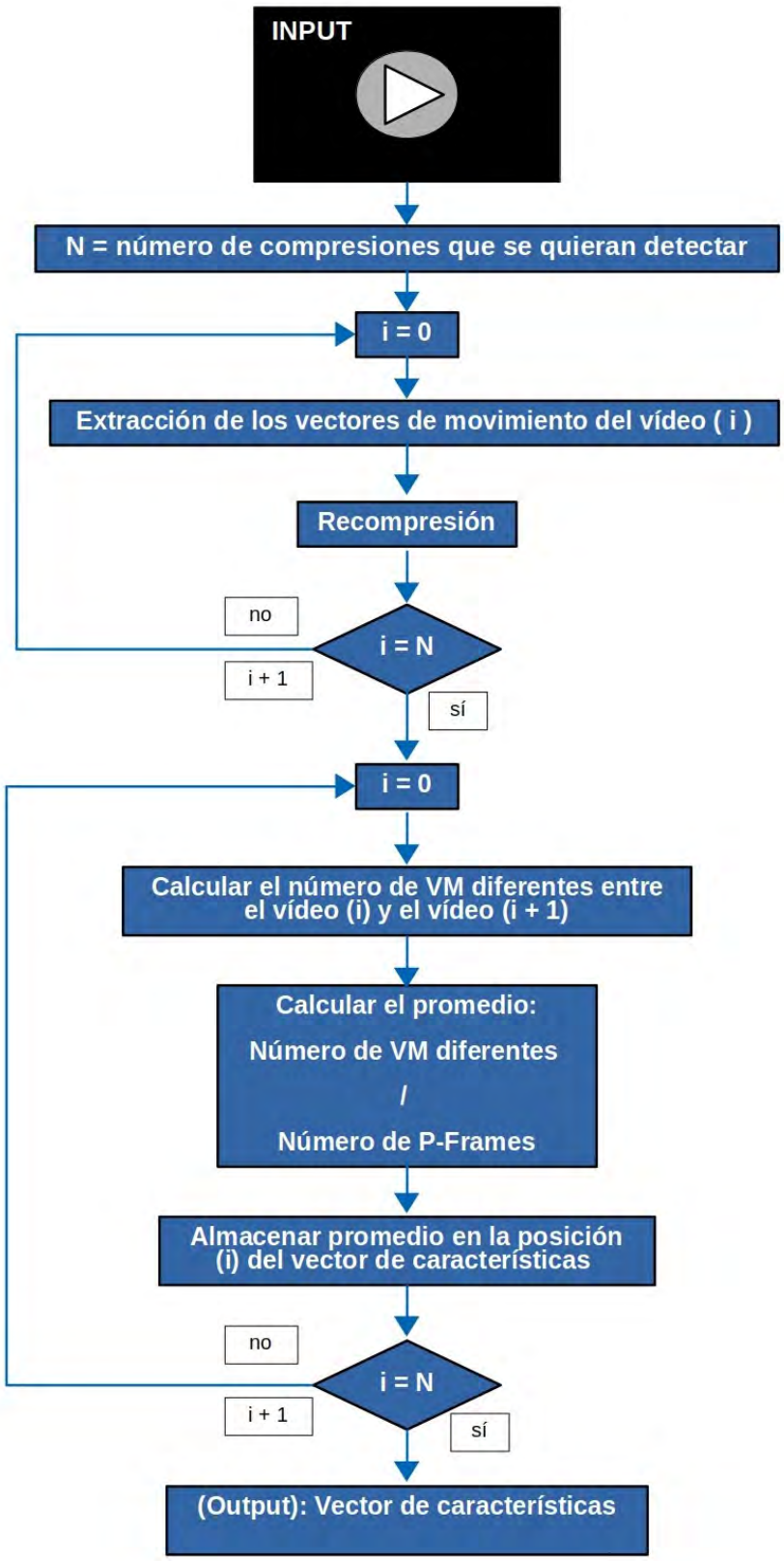


Figura 4.12: Diagrama del Algoritmo de Detección de Doble Compresión.

Capítulo 5

Experimentos y Resultados

Este capítulo tiene por objetivo describir los experimentos que se han llevado a cabo para evaluar la eficacia de los algoritmos desarrollados y los resultados obtenidos. El capítulo está dividido en dos secciones: Una dedicada al algoritmo de detección de empalme en imágenes, y la otra enfocada en el algoritmo de detección de recompresiones en vídeos. En cada una de estas secciones se va presentar la configuración del experimento realizado y su resultado obtenido.

En todos los experimentos realizados se ha utilizado *Python* como lenguaje de programación, debido a su gran flexibilidad para poder realizar el análisis de datos y su alta velocidad a la hora de gestionar tanto la entrada como la salida.

5.1. Evaluación del Algoritmo de Detección de Empalme en Imágenes

5.1.1. Configuración del Experimento

Para la evaluación de este algoritmo se ha hecho uso del dataset público CASIA v1.0([**?**]). Este dataset contiene imágenes manipuladas mediante operaciones de recorte y pegado utilizando Adobe Photoshop CS3 versión 10.0.1 en Windows XP. Las regiones empalmadas son de la misma imagen auténtica (copia-pega) o de otra imagen (empalme). Es por ello que en este experimento sólo van a participar aquellas imágenes del dataset que contengan la región empalmada procedente de una imagen diferente, ya que el algoritmo está diseñado para la detección empalme. También se ha generado un dataset especifico para este experimento con imágenes empalmadas manualmente de una alta resolución. La Tabla 5.1 muestra un resumen de las características del dataset utilizado en el experimento.

Las características del equipo con el que se han realizado los experimentos se presentan en la Tabla 5.2. Es un factor importante a tener en cuenta ya que los tiempos de ejecución de las diferentes pruebas varían según los recursos computaciones disponibles.

Tabla 5.1: Características del Dataset utilizado

Datasets	Formato	Resolución	Número de Imágenes
CASIA v1.0 [?]	JPEG	384x256	921 (empalme: 451)
Propio [?]	JPEG	1080x1920	30

Tabla 5.2: Características del equipo de experimentación

Recursos	Características
Sistema operativo	Ubuntu 18.04
Memoria	4 GB
Procesador	Intel® Core™ 2 Quad CPU Q8200 @ 2.33GHz x 4
Gráficos	NV96
Tipo de SO	64 bits
Disco	100 GB

5.1.2. Experimento

Este experimento está basado en la comprobación de las imágenes resultado de aplicar el algoritmo de detección de empalme en imágenes digitales. Este algortimo ha sido aplicado sobre las imágenes empalme del dataset CASIA v1.0 para su posterior revisión. También se ha utilizado un pequeño dataset propio con imágenes tomadas por IPHONE a las que se les ha aplicado empalme manualmente. La revisión consiste en comparar a simple vista si las regiones que más resaltan en la imagen resultado son las regiones que han sufrido el empalme.

En la Figura 5.1 se muestra un ejemplo positivo y en la Figura 5.2 se muestra uno negativo.

En la tabla 5.3 se muestran los resultados obtenidos.

Tabla 5.3: Detección de positivos tras aplicar el algoritmo

Datasets	Número de Imágenes	Positivos	Rendimiento
CASIA v1.0	451	248 / 55 %	00:00:25s
Propio	30	22 / 73.3 %	00:02:10s

(a) (b)

Figura 5.1: Ejemplo positivo: La zona de color blanco que más resalta en (b) corresponde con la región pegada en (a). El resto de pixeles blancos al estar aislados no se deben tener en cuenta.

5.2. Evaluación del Algoritmo de Detección de Recompresiones en Vídeos

5.2.1. Configuración del Experimento

Para evaluar este algoritmo se ha utilizado el dataset del grupo de trabajo de este TFG. El dataset contiene vídeos digitales procedentes de distintos modelos de dispositivos móviles con tamaños de resolución diferentes. Se han seleccionado vídeos en formato .MP4 con la resoluciones más comunes en estos momentos: 720x480, 720x1280, 1920x1080 y 3840x2160 (4K). La mayor parte de los vídeos seleccionados se han utilizado para el entrenamiento de la máquina de soporte vectorial con el fin de tener una base de conocimiento más amplia. El resto se han utilizado para el testing.

La Tabla 5.4 muestra un resumen de las características del dataset utilizado para el entrenamiento y la Tabla 5.5 muestra el empleado para el testing.

Tabla 5.4: Características del Dataset utilizado para el Entrenamiento

Resolución	Formato	Número de Vídeos
720x480	MP4	20
720x1280	MP4	36
1920x1080	MP4	36
4K	MP4	16

Las características del equipo en el cual se han realizado los experimentos se presentan en la Tabla 5.6. Es un factor importante a tener en cuenta ya que los tiempos de ejecución de las diferentes pruebas varían según los recursos computaciones disponibles.

Para la realización de los experimentos se ha utilizado un componente llamado

(a) (b)

Figura 5.2: Ejemplo negativo: Las zonas de píxels de color blanco en (b) no dejan distinguir de manera clara cual ha sido la zona pegada en (a).

Tabla 5.5: Características del Dataset utilizado para el Testing

Resolución	Formato	Número de Vídeos
720x480	MP4	10
720x1280	MP4	20
1920x1080	MP4	20
4K	MP4	9

MPEGflow [MPE] bajo licencia Massachusetts Institute of Technology (MIT) que sólo depende de FFMPEG. Esta herramienta se apoya en FFMPEG para facilitar las tareas de extracción de los vectores de movimiento de los frames de un vídeo y volcarlos en un fichero de texto.

También se ha utilizado el módulo LIBSVM, un software integrado que desempeña las funciones de una máquina de soporte vectorial, (C-SVC, nu-SVC), regresión (epsilon-SVR, nu-SVR) y estimación de distribución (SVM de una clase). Es compatible con la clasificación de clases múltiples.

Tabla 5.6: Características del equipo de experimentación

Recursos	Características
Sistema operativo	Ubuntu 18.04
Memoria	4 GB
Procesador	Intel® Core™ 2 Quad CPU Q8200 @ 2.33GHz x 4
Gráficos	NV96
Tipo de SO	64 bits
Disco	100 GB

5.2.2. Experimento

A lo largo de esta sección se muestran todos los experimentos que realizados para evaluar la efectividad de los algoritmos de identificación de manipulaciones basados en entrenamiento.

Este experimento pretende comprobar la variación de la precisión al aplicar el algoritmo sobre distintas resoluciones. Se estudia la capacidad de detectar si el vídeo es original, ha sido doblemente comprimdo o triplemente comprimido.

En primer lugar, hay que crear la base de conocimiento en la máquina de soporte vectorial, para lo cual los vídeos seleccionados para el entrenamiento del dataset de cada resolución se utilizan como entrada al algoritmo de detección de recompresiones.

Una vez entrenada la máquina con los ficheros de características generados por el algoritmo, se puede comenzar el testing.

Este testing consiste en extraer las características de los vídeos a testear para que la máquina, una vez entrenada, los clasifique en función de sus recompresiones.

Así, se han realizado dos experimentos: para el primero de ellos, el algoritmo extrae sólo dos características de los vídeos. La máquina de soporte vectorial dispondrá de las dos clases para discernir si el vídeo es original o ha sido recomprimido.

En el segundo experimento se extrae una característica más con el objetivo de poder determinar si ha sido recomprimido más de una vez, y en tal caso, cuantas veces (1 o 2).

Para cada uno de estos experimentos se han tomado los vectores de características escalados y sin escalar.

Los resultados de los experimentos para dos clases se encuentran en las tablas:

- Tabla 5.7: Muestra el porcentaje de confianza resultado del entrenamiento de la máquina de soporte vectorial para vectores de características escalados.
- Tabla 5.8: Resultados obtenidos del testing con vectores de características escalados.
- Tabla 5.9: Muestra el porcentaje de confianza resultado del entrenamiento de la máquina de soporte vectorial para vectores de características sin escalar.

- Tabla 5.10: Resultados obtenidos del testing con vectores de características sin escalar.
- Tabla 5.11: Resultados de rendimiento por resolución para un vídeo de similar duración de cada resolución.

Los resultados de los experimentos para tres clases se encuentran en las tablas:

- Tabla 5.12: Muestra el porcentaje de confianza resultado del entrenamiento de la máquina de soporte vectorial para vectores de características escalados.
- Tabla 5.13: Resultados obtenidos del testing con vectores de características escalados.
- Tabla 5.14: Muestra el porcentaje de confianza resultado del entrenamiento de la máquina de soporte vectorial para vectores de características sin escalar.
- Tabla 5.15: Resultados obtenidos del testing con vectores de características sin escalar.
- Tabla 5.16: Resultados de rendimiento por resolución para un vídeo de similar duración de cada resolución.

Tabla 5.7: Rate de confianza al aplicar el algoritmo de detección de recompresiones para datos escalados de 2 clases.

Resolución	**Rate**
720x480	92.5 %
720x1280	97.05 %
1920x1080	98.53 %
4K	100.0 %
MIX	91.8 %

Tabla 5.8: Variación de las precisiones al aplicar el algoritmo de detección de recompresiones para datos escalados de 2 clases.

Resolución	#	Original	Doble Compresión	Total
720x480	**Original**	**10/10**	0/10	100 %
	Doble Compresión	2/10	**8/10**	80 %
				18/20 (90.0 %)
720x1280	**Original**	**20/20**	0/20	100 %
	Doble Compresión	4/15	**11/15**	73.3 %
				31/35 (88.57 %)
1920x1080	**Original**	**19/20**	1/20	95 %
	Doble Compresión	0/15	**15/15**	100.0 %
				34/35 (97.14 %)
4K	**Original**	**9/9**	0/9	100.0 %
	Doble Compresión	0/8	**8/8**	100.0 %
				17/17 (100.0 %)
MIX	**Original**	**24/47**	23/47	51.0 %
	Doble Compresión	0/54	**54/54**	100.0 %
				78/101 (77.22 %)

Tabla 5.9: Rate de confianza al aplicar el algoritmo de detección de recompresiones para datos sin escalar de 2 clases.

Resolución	Rate
720x480	90.0 %
720x1280	88.2 %
1920x1080	98.5 %
4K	90.9 %
MIX	92.3 %

Tabla 5.10: Variación de las precisiones al aplicar el algoritmo de detección de recompresiones para datos sin escalar de 2 clases.

Resolución	#	Original	Doble Compresión	Total
720x480	**Original**	**9/10**	1/10	90 %
	Doble Compresión	1/10	**9/10**	90 %
				18/20 (90.0 %)
720x1280	**Original**	**20/20**	0/20	100 %
	Doble Compresión	0/15	**15/15**	100 %
				35/35 (100.0 %)
1920x1080	**Original**	**20/20**	0/20	100 %
	Doble Compresión	0/15	**15/15**	100 %
				35/35 (100.0 %)
4K	**Original**	**9/9**	0/9	100.0 %
	Doble Compresión	2/8	**5/8**	62.5 %
				14/17 (82.3 %)
MIX	**Original**	**42/47**	5/47	89.3 %
	Doble Compresión	4/54	**50/54**	92.6 %
				92/101 (91.1 %)

Tabla 5.11: Rendimiento tras aplicar el algoritmo de detección de recompresiones para 2 clases.

Resolución	Rendimiento
720x480	00:00:07.03s
720x1280	00:00:24.49s
1920x1080	00:01:16.32s
4K	00:05:44.13s

Tabla 5.12: Rate de confianza al aplicar el algoritmo de detección de recompresiones para datos escalados de 3 clases.

Resolución	Rate
720x480	70.0 %
720x1280	93.0 %
1920x1080	81.0 %
4K	94.0 %
MIX	73.87 %

Tabla 5.13: Variación de las precisiones al aplicar el algoritmo de detección de recompresiones para datos escalados de 3 clases.

Resolución	#	Original	Doble Compresión	Triple Compresión	Total
720x480	**Original**	**10/10**	0/10	0/10	100 %
	Doble Compresión	0/10	**5/10**	5/10	50 %
	Triple Compresión	0/10	0/10	**10/10**	100 %
Subtotal					**25/30 (83.33 %)**
720x1280	**Original**	**20/20**	0/20	0/20	100 %
	Doble Compresión	6/15	**9/15**	0/15	60 %
	Triple Compresión	1/15	8/15	**6/15**	40.0 %
					35/50 (70.0 %)
1920x1080	**Original**	**20/20**	0/20	0/20	100 %
	Doble Compresión	0/15	**0/15**	15/15	0.0 %
	Triple Compresión	0/15	1/15	**14/15**	93.3 %
					34/50 (68.0 %)
4K	**Original**	**9/9**	0/9	0/9	100.0 %
	Doble Compresión	0/8	**6/8**	2/8	75.0 %
	Triple Compresión	0/8	3/8	**5/8**	62.5 %
					20/25 (80.0 %)
MIX	**Original**	**25/47**	2/47	20/47	53.2 %
	Doble Compresión	0/54	**0/54**	54/54	0.0 %
	Triple Compresión	1/54	1/54	**52/54**	96.3 %
					77/155 (49.67 %)

Tabla 5.14: Rate de confianza al aplicar el algoritmo de detección de recompresiones para datos sin escalar de 3 clases.

Resolución	Rate
720x480	90.0 %
720x1280	76.0 %
1920x1080	78.0 %
4K	84.0 %
MIX	71.29 %

Tabla 5.15: Variación de las precisiones al aplicar el algoritmo de detección de recompresiones para datos sin escalar de 3 clases.

Resolución	#	Original	Doble Compresión	Triple Compresión	Total
720x480	**Original**	**10/10**	0/10	0/10	100 %
	Doble Compresión	1/10	**5/10**	4/10	50 %
	Triple Compresión	0/10	5/10	**5/10**	50 %
					20/30 (66.67 %)
720x1280	**Original**	**19/20**	1/20	0/20	95 %
	Doble Compresión	0/15	**12/15**	3/15	80 %
	Triple Compresión	0/15	2/15	**13/15**	86.67 %
					44/50 (88.0 %)
1920x1080	**Original**	**20/20**	0/20	0/20	100 %
	Doble Compresión	0/15	**8/15**	7/15	53.34 %
	Triple Compresión	0/15	8/15	**7/15**	46.67 %
					35/50 (70.0 %)
4K	**Original**	**7/9**	2/9	0/9	77.0 %
	Doble Compresión	0/8	**0/8**	8/8	0.0 %
	Triple Compresión	0/8	0/8	**8/8**	100 %
					15/25 (60.0 %)
MIX	**Original**	**44/47**	3/47	0/47	93.6 %
	Doble Compresión	6/54	**29/54**	19/54	53.7 %
	Triple Compresión	7/54	22/54	**25/54**	46.3 %
					98/155 (63.23 %)

Tabla 5.16: Rendimiento tras aplicar el algoritmo de detección de recompresiones para 3 clases.

Resolución	Rendimiento
720x480	00:00:16.23s
720x1280	00:00:55.11s
1920x1080	00:02:37.32s
4K	00:11:41.02s

Capítulo 6

Conclusiones y Trabajo Futuro

6.1. Conclusiones

El contenido de imágenes y vídeos digitales posee información que va más allá de la visual. Esta información es de gran valor forense, pues su correcta explotación puede garantizar la autenticidad e integridad del contenido. Debido a esto, las imágenes y vídeos digitales son una excepcional fuente de evidencias a la hora de resolver procesos judiciales.

El desarrollo y mejora continua de las nuevas tecnologías propicia que usuarios convencionales sean capaces de alterar el contenido de imágenes y vídeos con resultados profesionales, imperceptibles para el ojo humano. Ello se suma al hecho de que la detección de manipulaciones es una tarea compleja y también requiere de una mejora continua para adaptarse a tal escenario por lo que resulta imprescindible desarrollo de herramientas forenses capaces de detectar estas manipulaciones, cada vez más profesionales y habituales.

La línea de investigación que se ha seguido en este trabajo comienza realizando un estudio de las técnicas existentes de detección de manipulación sobre imágenes y vídeos digitales dedicando más esfuerzo a técnicas de detección de empalme en imágenes y en detección de doble compresión en vídeos.

Se han diseñado e implementado dos técnicas de detección de manipulación:

- En primer lugar, un algoritmo basado en el estándar de vídeo H.264/MPEG4 para la detección de recompresiones en vídeos MP4 que compara los vectores de movimiento de los macrobloques de dos compresiones secuenciales del mismo vídeo para, a continuación, hacer uso de una SVM que clasifique el vídeo.

- En segundo lugar, un algoritmo basado en la compresión JPEG para la detección de empalme en imágenes. Este algoritmo utiliza la técnica Error-level-Analysis y destaca aquellos píxeles con un nivel de compresión diferente. Finalmente, las áreas que no pertenecen al contenido original se marcan en la imagen.

Para la técnica de detección de empalme se ha utilizado el dataset público CASIA v1.0 y uno propio. Los resultados muestran que dependen directamente de la calidad de la

imagen, pues, para CASIA el algoritmo presenta dificultades a la hora de detectar la región que no pertenece a la imagen original. No obstante, el dataset propio contiene imágenes de alta resolución donde se obtuvo una precisión del 73.3 %. Es importante mencionar que la precisión del resultado la determina el investigador, pues es él el que tiene que considerar si en la imagen obtenida se encuentra resaltada de manera clara la zona empalmada. Por ello, es necesario conocer bien cómo funciona ELA. No obstante, Esta técnica sirve como primera evidencia, pues, al ser una técnica de detección de rápida respuesta puede servir como fuente de sospecha para un investigador con el fin de investigar más a fondo aquellas imágenes que han presentado regiones de color blanco sospechosas.

El dataset utilizado para evaluar la técnica de detección de recompresiones en vídeos ha sido el del grupo de trabajo GASS de la UCM. La evaluación ha constado de dos experimentos divididos en grupos según la resolución de cada vídeo:

- Detección de vídeo original o doblemente comprimido, el algoritmo ha conseguido una precisión máxima con datos escalados del 100 % para vídeos de resolución 4K, para el resto de resoluciones no baja del 90 %.

- Detección de vídeo original, doble comprimido, o triplemente comprimido donde la precisión disminuye ligeramente respecto a la detección de original o doble compresión, tiene un promedio de precisión del orden del 80 %. No obstante, el mejor resultado lo sigue teniendo una alta resolución.

Los experimentos se han realizado tanto para datos sin escalar como para datos escalados, obteniendo unos resultados muy similares entre ellos. Por tanto no es relevante hacer un escalado de los mismos.

También se han realizado pruebas mezclando todas las resoluciones obteniendo unos resultados menos precisos que en aquellas pruebas donde sí se han separando las resoluciones.

El rendimiento del algoritmo es directamente proporcional a la resolución del vídeo que se quiera procesar y a la cantidad de recompresiones que se quieran detectar.

6.2. Trabajo Futuro

Pese a los avances que se han generado con motivo de esta investigación, no podemos negar que nos encontramos ante un problema mucho más amplio del que no nos queda sino seguir investigando en el futuro, para resolver los problemas que la manipulación de fotografías y videos con fines maliciosos puedan generar. Así, en base a los resultados, las líneas futuras de investigación que se proponen en el presente trabajo son las siguientes:

- Extender el algoritmo de detección de recompresiones para utilizarlo con otros códecs de vídeo a parte del H264.

- Utilizar técnicas de deep learning y aumentar el número de características extraídas para mejorar la precisión de detección de recompresiones.
- Optimizar el algoritmo de detección de recompresiones para reducir el tiempo de procesado en vídeos de alta resolución.
- Extender el algoritmo de detección de empalme para detectar regiones empalmadas en vídeos.
- Extender el algoritmo de identificación de empalme para cuantificar los resultados con el fin de utilizar un clasificador SVM.

Bibliografía

[AB18] Javad Abbasi Aghamaleki and Alireza Behrad. Detecting double compressed mpeg videos with the same quantization matrix and synchronized group of pictures structure. *Journal of Electronic Imaging*, 27(1):013031, 2018.

[BBM+13] Paolo Bestagini, S Battaglia, Simone Milani, Marco Tagliasacchi, and Stefano Tubaro. Detection of temporal interpolation in video sequences. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 3033–3037. IEEE, 2013.

[BJGY10] X. Bo, W. Junwen, L. Guangjie, and D. Yuewei. Image Copy-Move Forgery Detection Based on SURF. In *2010 International Conference on Multimedia Information Networking and Security*, pages 889–892, Nanjing, China, November 2010.

[BMTT13] Paolo Bestagini, Simone Milani, Marco Tagliasacchi, and Stefano Tubaro. Local tampering detection in video sequences. In *Multimedia Signal Processing (MMSP), 2013 IEEE 15th International Workshop on*, pages 488–493. IEEE, 2013.

[CAGSO+13] J Rosales Corripio, David M. Arenas González, Ana Lucila Sandoval Orozco, Luis Javier García Villalba, Julio Hernandez-Castro, and Stuart James Gibson. Source smartphone identification using sensor pattern noise and wavelet transform. 2013.

[Che10] Girija Chetty. Blind and passive digital video tamper detection based on multimodal fusion. In *Proc. of the 14th WSEAS International Conference on Communications*, pages 109–117, 2010.

[CJS+16] Jieyuan Chen, Xinghao Jiang, Tanfeng Sun, Peisong He, and Shilin Wang. Detecting double mpeg compression with the same quantiser scale based on mbm feature. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 2064–2068. IEEE, 2016.

[CPF04] A. C Popescu and H. Farid. Exposing Digital Forgeries by Detecting Duplicated Image Regions. *Department of Computer Science*, 646, January 2004.

[DCG06] Asok De, Himanshu Chadha, and Sparsh Gupta. Detection of forgery in digital video. In *The 10th World Multi Conference on Systemics Cybernetics and Informatics*, volume 5, pages 229–233, 2006.

[DDSD12] Sreelekshmi Das, Gopu Darsan, L Shreyas, and Divya Devan. Blind detection method for video inpainting forgery. *International Journal of Computer Applications*, 60(11), 2012.

[elaa] Error-level-analysisis.

[elab] Explicación de los modelos de color.

[elac] Fotoforensics.

[Far99] Hany Farid. Detecting digital forgeries using bispectral analysis. 1999.

[FFMa] Debug/macroblocksandmotionvectors.

[FFMb] Ffmpeg.

[FSL03] J. Fridrich, D. Soukal, and J. Lukas. Detection of Copy Move Forgery in Digital Images. In *Proceedings of the Digital Forensic Research Workshop*, pages 5–8, Binghamton, New York, August 2003.

[FSS06] Dongdong Fu, Yun Q Shi, and Wei Su. Detection of image splicing based on hilbert-huang transform and moments of characteristic functions with wavelet decomposition. In *International workshop on digital watermarking*, pages 177–187. Springer, 2006.

[GC11] Julian Goodwin and Girija Chetty. Blind video tamper detection based on fusion of source features. In *Digital Image Computing Techniques and Applications (DICTA), 2011 International Conference on*, pages 608–613. IEEE, 2011.

[GFB+14] Alessandra Gironi, Marco Fontani, Tiziano Bianchi, Alessandro Piva, and Mauro Barni. A video forensic technique for detecting frame deletion and insertion. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 6226–6230. IEEE, 2014.

[GHK+17] Teddy Surya Gunawan, Siti Amalina Mohammad Hanafiah, Mira Kartiwi, Nanang Ismail, Nor Farahidah Za'bah, and Anis Nurashikin Nordin. Development of photo forensics algorithm by detecting photoshop manipulation using error level analysis. *Indonesian Journal of Electrical Engineering and Computer Science*, 7(1):131–137, 2017.

[HC06] Yu-Feng Hsu and Shih-Fu Chang. Detecting image splicing using geometry invariants and camera characteristics consistency. In *Multimedia and Expo, 2006 IEEE International Conference on*, pages 549–552. IEEE, 2006.

[HGZ08] H. Huang, W. Guo, and Y. Zhang. Detection of Copy-Move Forgery in Digital Images Using SIFT Algorithm. In *2008 IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application*, volume 2, pages 272–276, December 2008.

[HRLL13] Dai-Kyung Hyun, Seung-Jin Ryu, Hae-Yeoun Lee, and Heung-Kyu Lee. Detection of upscale-crop and partial manipulation in surveillance video based on sensor pattern noise. *Sensors*, 13(9):12605–12631, 2013.

[JBdSC17] Daniel Cavalcanti Jeronymo, Yuri Cassio Campbell Borges, and Leandro dos Santos Coelho. Image forgery detection by semi-automatic wavelet soft-thresholding with error level analysis. *Expert Systems with Applications*, 85:348–356, 2017.

[JHS+18] Xinghao Jiang, Peisong He, Tanfeng Sun, Feng Xie, and Shilin Wang. Detection of double compression with the same coding parameters based on quality degradation mechanism analysis. *IEEE Transactions on Information Forensics and Security*, 13(1):170–185, 2018.

[JWS+13] Xinghao Jiang, Wan Wang, Tanfeng Sun, Y.Q. Shi, and Shilin Wang. Detection of double compression in mpeg-4 videos based on markov statistics. 20:447–450, 05 2013.

[KOS10] Michihiro Kobayashi, Takahiro Okabe, and Yoichi Sato. Detecting forgery from static-scene video based on inconsistency in noise level functions. *IEEE Transactions on Information Forensics and Security*, 5(4):883–892, 2010.

[LCDG11] Qiguang Liu, Xiaochun Cao, Chao Deng, and Xiaojie Guo. Identifying image composites through shadow matte consistency. *IEEE Transactions on Information Forensics and Security*, 6(3):1111–1122, 2011.

[LG06] Aaron Langille and Minglun Gong. An efficient match-based duplication detection algorithm. In *Computer and Robot Vision, 2006. The 3rd Canadian Conference on*, pages 64–64. IEEE, 2006.

[LHQ06] Weiqi Luo, Jiwu Huang, and Guoping Qiu. Robust detection of region-duplication forgery in digital image. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 4, pages 746–749. IEEE, 2006.

[LS09] Qingzhong Liu and Andrew H Sung. A new approach for jpeg resize and image splicing detection. In *Proceedings of the First ACM workshop on Multimedia in forensics*, pages 43–48. ACM, 2009.

[LT14] Cheng-Shian Lin and Jyh-Jong Tsay. A passive approach for effective detection and localization of region-level video forgery with spatio-temporal coherence analysis. *Digital Investigation*, 11(2):120–140, 2014.

[LWTS07] Guohui Li, Qiong Wu, Dan Tu, and Shaojie Sun. A sorted neighborhood approach for detecting duplicated regions in image forgeries based on dwt and svd. In *Multimedia and Expo, 2007 IEEE International Conference on*, pages 1750–1753. IEEE, 2007.

[MCP+07] N Mondaini, Roberto Caldelli, Alessandro Piva, Mauro Barni, and Vito Cappellini. Detection of malevolent changes in digital video for forensic applications. In *Security, steganography, and watermarking of multimedia contents IX*, volume 6505, page 65050T. International Society for Optics and Photonics, 2007.

[MPE] github: vadimkantorov/mpegflow.

[MVP07] AN Myna, MG Venkateshmurthy, and CG Patil. Detection of region duplication forgery in digital images using wavelets and log-polar mapping. In *Conference on Computational Intelligence and Multimedia Applications, 2007. International Conference on*, volume 3, pages 371–377. IEEE, 2007.

[NC04] T-T Ng and S-F Chang. A model for image splicing. In *Proceedings of the International Conference on Image Processing, 2004*, volume 2, pages 1169–1172. IEEE, 2004.

[Nie] Eva Martín Nieto. The value of photography: Anthropology and image.

[Nor02] José Luis Sánchez Noriega. *Historia del cine: teoría y géneros cinematográficos, fotografía y televisión*. Anaya-Spain, 2002.

[NWW17] Sophie J Nightingale, Kimberley A Wade, and Derrick G Watson. Can people identify original and manipulated photos of real-world scenes? *Cognitive research: principles and implications*, 2(1):30, 2017.

[PSSA14] Ramesh Chand Pandey, Sanjay Kumar Singh, KK Shukla, and Rishabh Agrawal. Fast and robust passive copy-move forgery detection using surf and sift image features. In *Industrial and Information Systems (ICIIS), 2014 9th International Conference on*, pages 1–6. IEEE, 2014.

[SCC07] Yun Q Shi, Chunhua Chen, and Wen Chen. A natural image model approach to splicing detection. In *Proceedings of the 9th workshop on Multimedia & security*, pages 51–62. ACM, 2007.

[Sny09] Ian Snyder. Jpeg image compression, 2009.

[SNZ11] Yuting Su, Weizhi Nie, and Chengqian Zhang. A frame tampering detection algorithm for mpeg videos. In *Information Technology and Artificial Intelligence Conference (ITAIC), 2011 6th IEEE Joint International*, volume 2, pages 461–464. IEEE, 2011.

[SOAGC+14] Ana Lucila Sandoval Orozco, David M. Arenas González, J Rosales Corripio, LJ García Villalba, and Julio C Hernandez-Castro. Source identification for mobile devices, based on wavelet transforms combined with sensor imperfections. *Computing*, 96(9):829–841, 2014.

[SWJ12] Tanfeng Sun, Wan Wang, and Xinghao Jiang. Exposing video forgeries by detecting mpeg double compression. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pages 1389–1392. IEEE, 2012.

[SX10] Yuting Su and Junyu Xu. Detection of double-compression in mpeg-2 videos. In *Intelligent Systems and Applications (ISA), 2010 2nd International Workshop on*, pages 1–4. IEEE, 2010.

[WBI+14] Ainuddin Wahid Abdul Wahab, Mustapha Aminu Bagiwa, Mohd Yamani Idna Idris, Suleman Khan, Zaidi Razak, and Muhammad Rezal Kamel Ariffin. Passive video forgery detection techniques: a survey. In *Information assurance and security (IAS), 2014 10th International Conference on*, pages 29–34. IEEE, 2014.

[WDT] Anil Dada Warbhe, Rajiv V Dharaskar, and Vilas M Thakare. International journal of engineering sciences & research technology block based image forgery detection techniques.

[WF06] Weihong Wang and Hany Farid. Exposing digital forgeries in video by detecting double mpeg compression. In *Proceedings of the 8th workshop on Multimedia and security*, pages 37–47. ACM, 2006.

[WF07a] Weihong Wang and Hany Farid. Exposing digital forgeries in interlaced and deinterlaced video. *IEEE Transactions on Information Forensics and Security*, 2(3):438–449, 2007.

[WF07b] Weihong Wang and Hany Farid. Exposing digital forgeries in video by detecting duplication. In *Proceedings of the 9th workshop on Multimedia & security*, pages 35–42. ACM, 2007.

[WF09] Weihong Wang and Hany Farid. Exposing digital forgeries in video by detecting double quantization. In *Proceedings of the 11th ACM workshop on Multimedia and security*, pages 39–48. ACM, 2009.

[WIWS15] Nor Bakiah Abd Warif, Mohd Yamani Idna Idris, Ainuddin Wahid Abdul Wahab, and Rosli Salleh. An evaluation of error level analysis in image forensics. In *System Engineering and Technology (ICSET), 2015 5th IEEE International Conference on*, pages 23–28. IEEE, 2015.

[WSBL03] Thomas Wiegand, Gary J Sullivan, Gisle Bjontegaard, and Ajay Luthra. Overview of the h. 264/avc video coding standard. *IEEE Transactions on circuits and systems for video technology*, 13(7):560–576, 2003.

[WWZ11] Qiumin Wu, Shuozhong Wang, and Xinpeng Zhang. Log-polar based scheme for revealing duplicated regions in digital images. *IEEE Signal Processing Letters*, 18(10):559–562, 2011.

[XSY12] Junyu Xu, Yuting Su, and Xingang You. Detection of video transcoding for digital forensics. In *Audio, Language and Image Processing (ICALIP), 2012 International Conference on*, pages 160–164. IEEE, 2012.

[XYL+17] Min Xia, Gaobo Yang, Leida Li, Ran Li, and Xingming Sun. Detecting video frame rate up-conversion based on frame-level analysis of average texture variation. *Multimedia Tools and Applications*, 76(6):8399–8421, 2017.

[YYSL16] Yuxuan Yao, Gaobo Yang, Xingming Sun, and Leida Li. Detecting video frame-rate up-conversion based on periodic properties of edge-intensity. *Journal of Information Security and Applications*, 26:39–50, 2016.

[ZCQ+10] Wei Zhang, Xiaochun Cao, Yanling Qu, Yuexian Hou, Handong Zhao, and Chenyang Zhang. Detecting and extracting the photo composites using planar homography and graph cut. *IEEE transactions on information forensics and security*, 5(3):544–555, 2010.

[ZCZ+09] Wei Zhang, Xiaochun Cao, Jiawan Zhang, Jigui Zhu, and Ping Wang. Detecting photographic composites using shadows. In *Multimedia and Expo, 2009. ICME 2009. IEEE International Conference on*, pages 1042–1045. IEEE, 2009.

[ZKR08] Zhen Zhang, Jiquan Kang, and Yuan Ren. An effective algorithm of image splicing detection. In *Computer Science and Software Engineering, 2008 International Conference on*, volume 1, pages 1035–1039. IEEE, 2008.

[ZLLW10] Xudong Zhao, Jianhua Li, Shenghong Li, and Shilin Wang. Detecting digital image splicing in chroma spaces. In *International Workshop on Digital Watermarking*, pages 12–22. Springer, 2010.